THE INTRODUCTION OF DIFFERENTIATION INTO THE FORTRAN 63 LANGUAGE

RAYMOND L. WILLIAMS

THE INTRODUCTION OF DIFFERENTIATION INTO THE FORTRAN 63 LANGUAGE

* * * * *

Raymond L. Williams

and

Donald L. Katz

THE INTRODUCTION OF
DIFFERENTIATION
INTO THE
FORTRAN 63 LANGUAGE

by

Raymond L. Williams

Lieutenant Commander, Supply Corps, United States Navy

and

Donald L. Katz

Lieutenant, United States Navy

Submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE
IN
MANAGEMENT (DATA PROCESSING)

United States Naval Postgraduate School
Monterey, California

1 9 6 5

THE INTRODUCTION OF DIFFERENTIATION INTO THE FORTRAN 63 LANGUAGE

by

Raymond L. Williams

and

Donald L. Katz

This work is accepted as fulfilling

the thesis requirements for the degree of

MASTER OF SCIENCE

IN

MANAGEMENT (DATA PROCESSING)

from the

United States Naval Postgraduate School

ABSTRACT

A method is described for the introduction into the Fortran 63 language of a special operator for differentiation which would be executed when encountered during compilation. The original expression would be extracted from the Fortran equation and replaced by its symbolic derivative; the translation process would then proceed in the normal manner.

A model algebraic translator, written in Fortran 63, is described and test results are presented.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

I. INTRODUCTION

Hanson[1] and others before him recognized the need for a compiler routine which would accept an algebraic expression and print-out a symbolic representation of the derivative of that expression in addition to generating the necessary machine symbolic macros required for subsequent numerical evaluation by the computer. This thesis presents a method to accomplish those goals in the Fortran 63 language. [3]

The method described herein assumes the utilization of reverse Polish notation[2] and a left to right scan during the compilation process.

"DX" is defined as a special operator, signifying differentiation. This operator is somewhat analogous in design and use to the normal Fortran "Function", e.g., SINF, which appears to be treated both as an operator and an operand during its life cycle. Certainly, SINF signifies and initiates an operation. Yet, during the translation process, the function is treated as an operand and is paired with a special operator,

[1] Hanson, J. W., J. S. Caviness and C. Joseph. "Analytic Differentiation by Computer", Communications, ACM, Vol. 5, No. 6, June 1962: 349.

[2] Hamblin, C. L., "Translation to and from Polish notation", Computer Journal 5, 3, October 1962: 210-213.

usually the period symbol. This similarity in the utilization of the "DX" will be evident. However, since there will also be dissimilarities, the authors have chosen to refer to the differentiation operator, DX, as a "dual-operator" for clarity of exposition.

The routine for translation from Fortran notation to reverse Polish notation requires the recognition of the dual-operator, DX, and a special subroutine for translation of the dual-operator and its operand. For the purpose of describing this procedure, a simplified routine for translation to reverse Polish is presented.

The next operation of translation of reverse Polish notation into machine symbolic language, likewise, requires recognition of the dual-operator, DX, and a special subroutine for execution of the derivative operator. This subroutine is designed to extract the derivative operator and its operand from the Polish string, execute the differentiation, place the symbolic result into the Polish string, and return control to the primary routine at the original point of entry.

II. HISTORICAL BACKGROUND

Hanson[3] presented a program which would accomplish the analytic differentiation of a mathematical expression on a digital

[3] Hanson, J. W., et al., op. cit., 349-355.

computer. He first described the methods and accomplishments which had taken place in this field prior to the writing of his paper. His approach was to take an input mathematical expression and covert it into triples. He then took the differentiated expression of each of these triples incorporating the Ershov algorithm[4] in the differentiation.

In February 1964, R. E. Wengert[5] wrote a paper which developed a program to give a numerical evaluation of a derivative without developing the analytic expression. His method was to set up a library of elementary function subroutines. This would work only for the simple routines, and would involve the programmer in using the CALL statements in his program to arrive at the solution to his problem. Wilkens[6] and Bellman[7] also propose the use of library subroutines to determine the value of a derivative.

[4]Ibid.

[5]Wenger, R. E., "A Simple Automatic Derivative Evaluation Program", Communications, ACM, Vol. 7, No. 8, August 1964: 463-464.

Wilkins, R. D., "Investigation of a New Analytical Method for Numerical Derivative Evaluation", Communications, ACM, Vol. 7, No. 8, August 1964: 465-471.

[7]Bellman, R. E., H. Kagiwada, R. E. Kalaba; "Wengert's Numerical Method for Partial Derivatives, Orbit Determination and Quasilinearization", Communications, ACM, Vol. 8, No. 4, April 1965: 231-232.

Slagle[8] and several others who have written on the subject have used list processing to accomplish the taking of the derivative. Of course, this process cannot be used by any computer which does not have the list processing capability. However, for additional comment on the disadvantages of this process, see Smith.[9] The paper by Peter J. Smith was received too late for a true evaluation by the authors. However, he proposed differentiating in symbolic form without the use of list processing, or the requiring of the programmer to write subroutines as a part of the program. There is, however, no place in Smith's paper any mention of getting a numerical answer to the problem as well as the symbolic representation of the derivative.

A means is needed whereby the symbolic representation of the derivative is available, as well as the numerical value for that derivative.

[8]Smith, P. J., Symbolic Derivatives without List Processing, Subroutines, or Recursion, Ballistic Research Laboratories Memorandum Report No. 1630, February 1965.

[9]*Ibid*.

III. METHOD

1. Definition of an operator for differentiation.

An operator usually must perform two functions: identify its operands and execute its designed operation upon those operands. In addition, a derivative operator must identify the variable(s) of differentiation. To facilitate the performance of those functions, the authors propose the adaptation of the dual nature accorded the "Function" within the Fortran language.

For use in the Fortran language, DX is designated as an operator for differentiation. The scope of the DX operator, i.e., its variables of differentiation and its expression to be differentiated, are assigned by enclosure in parentheses in accordance with the following format: DX; the left parenthesis; the list of variables of differentiation, each separated by a comma, and the list terminated by a double period; the expression to be differentiated; the right parenthesis. For example, the algebraic equation,

$$A = X \frac{\partial^2}{\partial X\, \partial Y} (X^2 Y^2 + \frac{\partial}{\partial Z} (XY\, SIN^2 Z)),$$

would be expressed in Fortran 63 as follows:

```
A = X * DX(X,Y..X**2 * Y**2 + DX(Z..X * Y * SIN(Z)**2))
```

2. Translation of the Derivative Dual-Operator and its argument into reverse Polish notation.

The stack compilation technique described by Wegner[10] is utilized in the transformation process.

In order to determine its "scope" in a reverse Polish string, the DX operator must assume the dual nature afforded the Fortran "Function". Therefore, the DX is assigned a suffix operator; namely, a double comma (the obvious "prime" not being available on a keyboard). When DX is encountered during translation from Fortran into reverse Polish, control passes to the DX closed subroutine (Figure 4 Page 23) which performs the following functions. The "DX" is released immediately into the Polish string, and its suffix operator is placed into the operator shunt with a priority assigned sufficiently high to insure its release immediately upon the encounter of the first operator following the expression being differentiated. The argument's left parenthesis is then placed into the shunt, the variables of differentiation extracted and placed into the Polish string, the

[10]Wegner, P., Introduction to System Programming, Academic Press, 1964: 100-121.

double period placed into the Polish string, and control then returned to the superior routine for normal processing of the remainder of the expression.

3. Execution of the Derivative Dual-Operator.

Upon encountering the DX dual-operator during the translation of the reverse Polish notation into a machine symbolic language, control passes to the DX closed subroutine. (Figure 5)

First, the variables of differentiation are extracted, indexed, and stored. The double period signifies the beginning of the expression to be differentiated. The differentiation process is then executed with respect to the variables in the order listed by the programmer.

Differentiation is executed upon operands, whether single symbols or expressions, in the order of presentation of their algebraic operators, in the normal left to right scan. A simple algorithm (Page 27) is employed to identify the operands of each algebraic operator encountered. The operands are identified with respect to position and complexity, and their derivatives are identified as zero or otherwise. The differentiation process is then executed in accordance with the applicable rule (Page 28). The partial result thus obtained is stored in an array, which

serves as a base upon which to build the derivative of the following algebraic operator and its pair of operands. In most cases, this partial result becomes the first part of the derivative of the following operation; thus symbol manipulation is minimized. In the remaining few cases only part of the partial result requires shifting. Further, zeroes are identified immediately, eliminated, and the process scan proceeds on to the next algebraic operator.

Upon completion of execution of partial differentiation, the symbolic result is patched into the reverse Polish string between the DX and its suffix. At this point, the expression differentiated and its derivative can be printed in reverse Polish notation or transformed into normal form prior to print-out. Though not provided in the attached model program, a code word could easily be inserted into the DX format, following the variables of differentiation, where it could easily be identified and used to command a print-out of the symbolic derivative only when required by the programmer for analysis.

If a DX operator is encountered during differentiation, the partially processed results are stored, the new DX operator is executed, the stored partial results retrieved, and the process continued from the point of interruption.

IV. TEST RESULTS

Sample runs of the program were made on the CDC 1604 computer at the U. S. Naval Postgraduate School. Thirty test equations were used, the last of which being the equation presented by Hanson.[11]

The time required to run the entire program, including the printing out of the original input equation in Fortran, the translation of the Fortran equation to reverse Polish notation, the derivatives of the required portions of the equation in reverse Polish notation, and the printing out of the macros for the evaluation of the equation was five minutes, fifty-three seconds. The time required for the compilation of the program was three minutes, five seconds.

Additional runs of the program were made to determine the time for the internal execution of the program, eliminating all print-outs. The time for this run was four minutes. The time required to execute the program giving all print-outs with the exception of the macros was four minutes, twenty-eight seconds.

The sample programming runs that were made showed that thirty equations, some of which required the derivative of a derivative, required two minutes, forty-eight seconds.

[11] Hanson, J. W., et al., op. cit., 349-355.

BIBLIOGRAPHY

1. Bellman, R. E., H. Kagiwada, and R. E. Kalaba, "Wengert's Numerical Method for Partial Derivatives, Orbit Determination and Quasilinearization", Communication, ACM, Vol. 8, No. 4, April 1965: 231-232.

2. Bellman, R. E., J. D. Buell, R. E. Kalaba, "Numerical Integration of a Differential-Difference Equation with a Decreasing Time-Lag", Communications, ACM, Vol. 8, No. 4, April 1965: 227-228.

3. Control Data 1604/1604A Computer Fortran 63/Reference Manual, Pub. No. 60052900, Revision A, Control Data Corporation.

4. Hamblin, C. L., "Translation to and from Polish Notation", Computer Journal 5, 3, October 1962: 210-213.

5. Hanson, J. W., J. S. Caviness, and C. Joseph, "Analytic Differentiation by Computer", Communications, ACM, Vol. 5, No. 6, June 1962: 349-355.

6. Smith, P. J., Symbolic Derivatives Without List Processing, Subroutines, or Recursion, Ballistic Research Laboratories Memorandum Report No. 1630, February 1965.

7. Wilkins, R. D., "Investigation of a New Analytical Method for Numerical Derivative Evaluation", Communications, ACM, Vol. 7, No. 8, August 1964: 465-471.

8. Wegner, P., Introduction to System Programming, Academic Press, London and New York 1964.

9. Wengert, R. E., "A Simple Automatic Derivative Evaluation Program", Communications, ACM, Vol. 7, No. 8, August 1964: 463-464.

[illegible] Magazine",
[illegible]).

[illegible] Analytic
[illegible] Communications ACM, Vol. 8,
[illegible]

[illegible] Processing,
[illegible] Information
[illegible] No. 1151, February 1958.

[illegible] "Considerations of a New Analytical Method
[illegible]", Communications ACM,
Vol. 7, No. 8, August 1964, [illegible].

[illegible] Academic
Press, London and New York, 1964.

[illegible] "A Simple [illegible] Evaluation
Program", Communications ACM, Vol. 7, No. 8, August 1964,
[illegible].

APPENDIX

LIST OF DIFFERENTIALS PROVIDED IN MODEL PROGRAM

$d\ (a * u)$	$= a * du$
$d\ (u + v)$	$= du + dv$
$d\ (u * v)$	$= u * dv + v * du$
$d\ (u / v)$	$= (v * du - u * dv) / v^2$
$d\ (u^n)$	$= n * u^{n-1} * du$
$d\ (u^v)$	$= v * u^{v-1} * du + u^v * \ln(u) * dv$
$d\ (a^u)$	$= a^u * \ln(a) * du$
$d\ (u^u)$	$= u^u * (1 + \ln(u)) * du$
$d\ \exp(u)$	$= \exp(u) * du$
$d\ \ln(u)$	$= du/u$
$d\ \sin(u)$	$= \cos(u) * du$
$d\ \cos(u)$	$= -\sin(u) * du$
$d\ \tan(u)$	$= \sec^2 (u) * du$
$d\ \mathrm{sqrt}(u)$	$= .5*du/ \mathrm{sqrt}(u)$

Note 1. Additional functions can be easily added to routine.

Figure 1

INDEX FOR REVERSE POLISH ROUTINE

I	POSITION INDEX FOR ALPHA
K	POSITION INDEX FOR CHARACTR
L	POSITION INDEX FOR REVERSE POLISH STRING
M	POSITION INDEX FOR OPERATOR SHUNT
N	INDEX FOR OPERATOR CODE
ALPHA(I)	INPUT ALGEBRAIC EXPRESSION.
CHARACTR(K)	ASSEMBLY ARRAY FOR THE CHARACTERS OF A SYMBOL.
SHUNT(M)	SHUNT FOR OPERATORS.
PSHUNT(M)	PRIORITY OF OPERATORS IN SHUNT.
PARITY	PARITY COUNTER FOR OPERANDS AND OPERATORS (LESS PARENS)
Z	PARITY COUNTER FOR PARENS
TERMINUS	END POSITION OF THE REVERSE POLISH STRING

Figure 2

INDEX FOR DERIVATIVE ROUTINE

N	INDEX FOR OPERATOR CODE
ND	POSITION INDEX FOR DERIVATIVE STRING
NE	POSITION INDEX FOR EXPRESSION STRING
NL	COUNTER FOR LEVELS (AN UNBROKEN SEQUENCE OF OPERANDS)
NN	INDEX FOR FUNCTIONS
NO	COUNTER FOR NUMBER OF OPERANDS IN AN UNBROKEN SEQUENCE
NP	POSITION INDEX FOR REVERSE POLISH STRING
NS	POSITION INDEX FOR OPERANDS SHUNT
NV	COUNTER FOR VARIABLES OF DIFFERENTIATION
NDX	COUNTER FOR DUAL-OPERATORS IN PROCESS
NIV	INDEX FOR PARTIAL DERIVATIVES.
A	SINGLE OPERAND
B	SINGLE OPERAND
C	CONSTANT
D	DERIVATIVE
E	EXPRESSION BEING DIFFERENTIATED
V	VARIABLE
DX	DUAL-OPERATOR FOR DIFFERENTIATION
OP	OPERATORS
POL	REVERSE POLISH STRING
BEGIND	BEGINNING OF DERIVATIVE STRING
BEGINE	BEGINNING OF EXPRESSION STRING
ENDD	END OF DERIVATIVE STRING
ENDE	END OF EXPRESSION STRING
LAM	LEFT SIDE EXPRESSION
LAMP	DERIVATIVE OF LAMBDA
RHO	RIGHT SIDE EXPRESSION
RHOP	DERIVATIVE OF RHO
SHU	SHUNT FOR OPERANDS
SING	SINGLE OPERANDS IN AN UNBROKEN STRING WAITING FOR OPERATOR
VARS	VARIABLES
FUNCTION	
ENTRANCE	ENTRY POSITION INTO REVERSE POLISH STRING (POSTION OF DX)
EXIT	EXIT POSITION FROM REVERSE POLISH STRING (POSITION OF ,,)
TERMINUS	END POSITION OF REVERSE POLISH STRING

Figure 3

TRANSFORMATION OF FORTRAN 63 ALGEBRAIC EQUATION
INTO REVERSE POLISH NOTATION

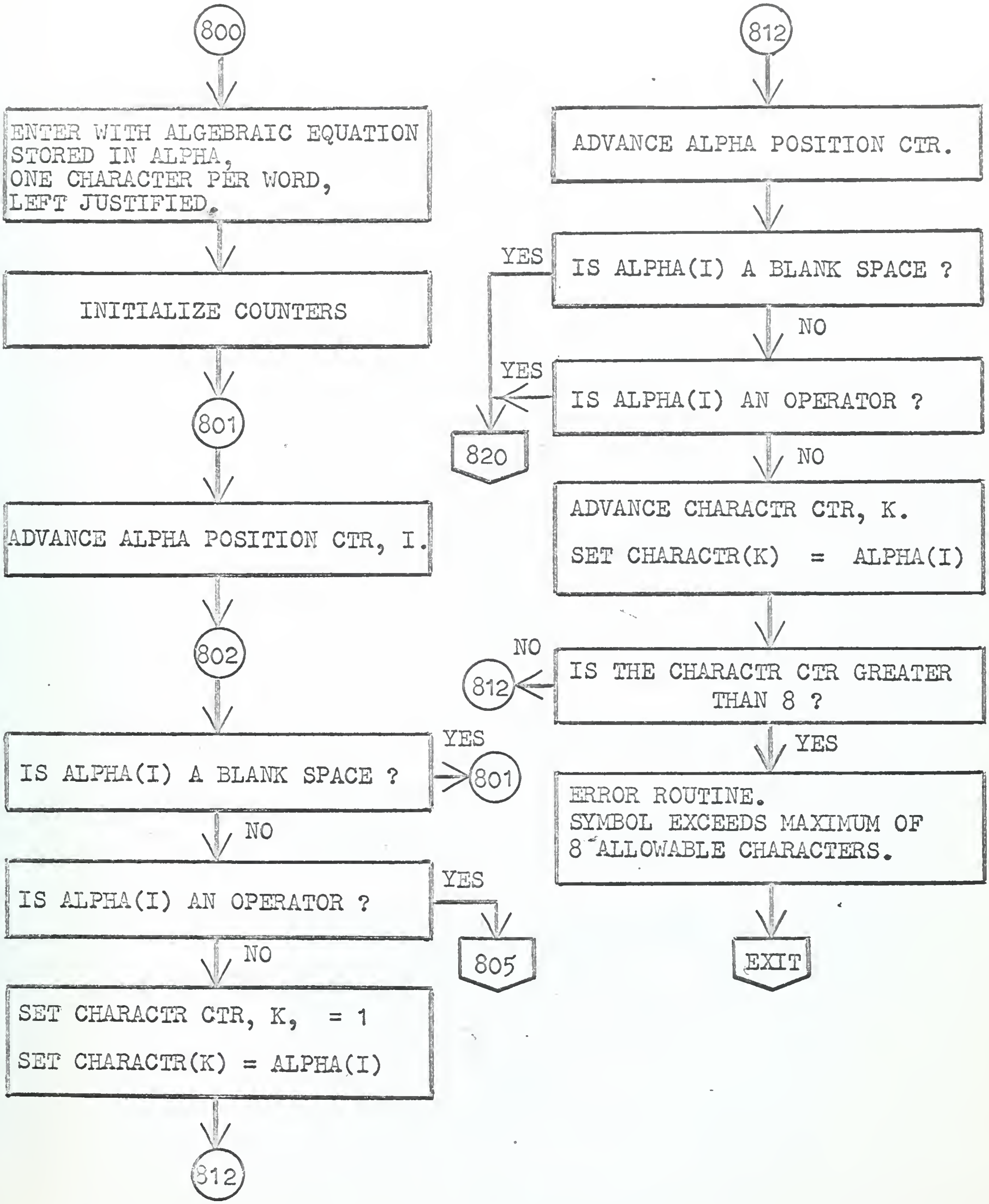


Figure 4

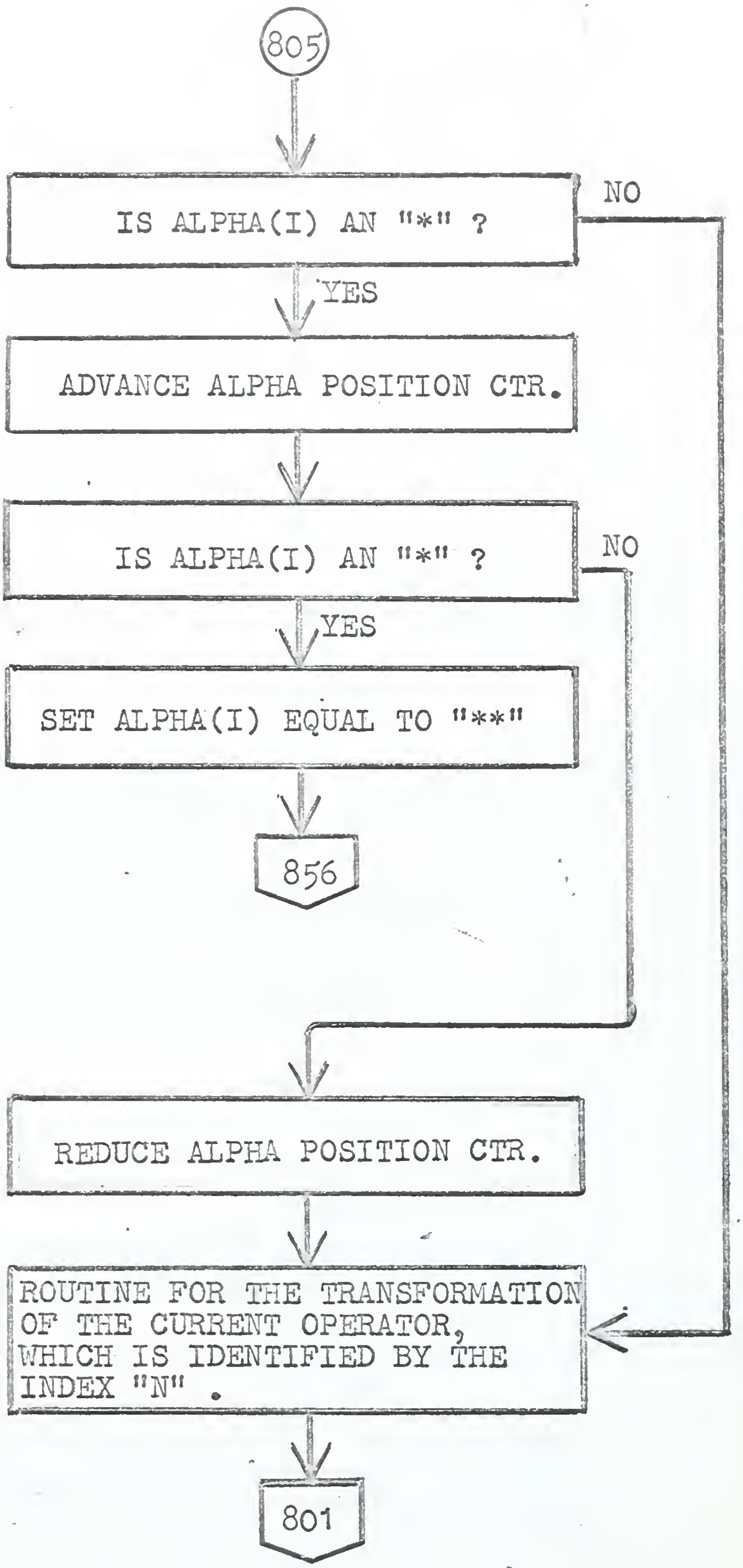


Figure 4 (Cont'd)

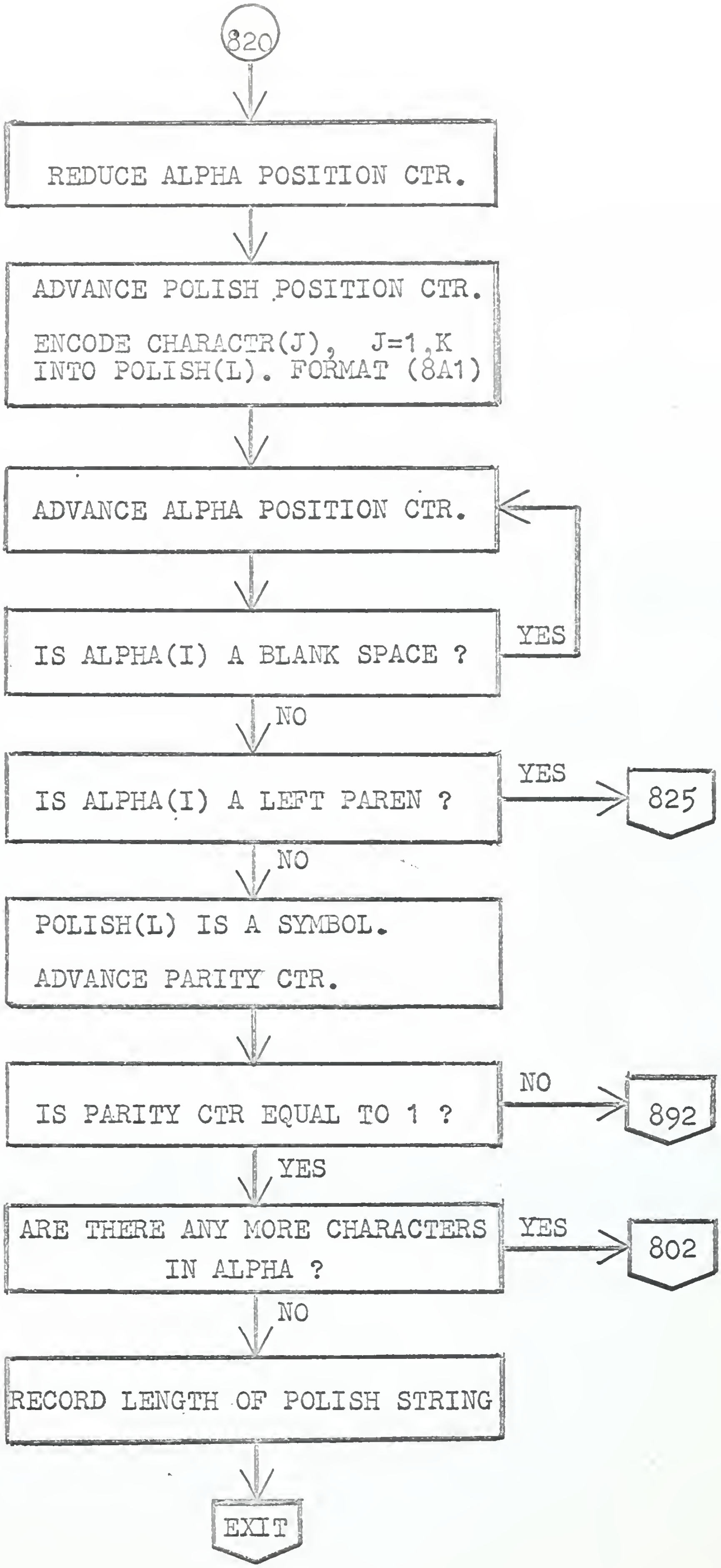
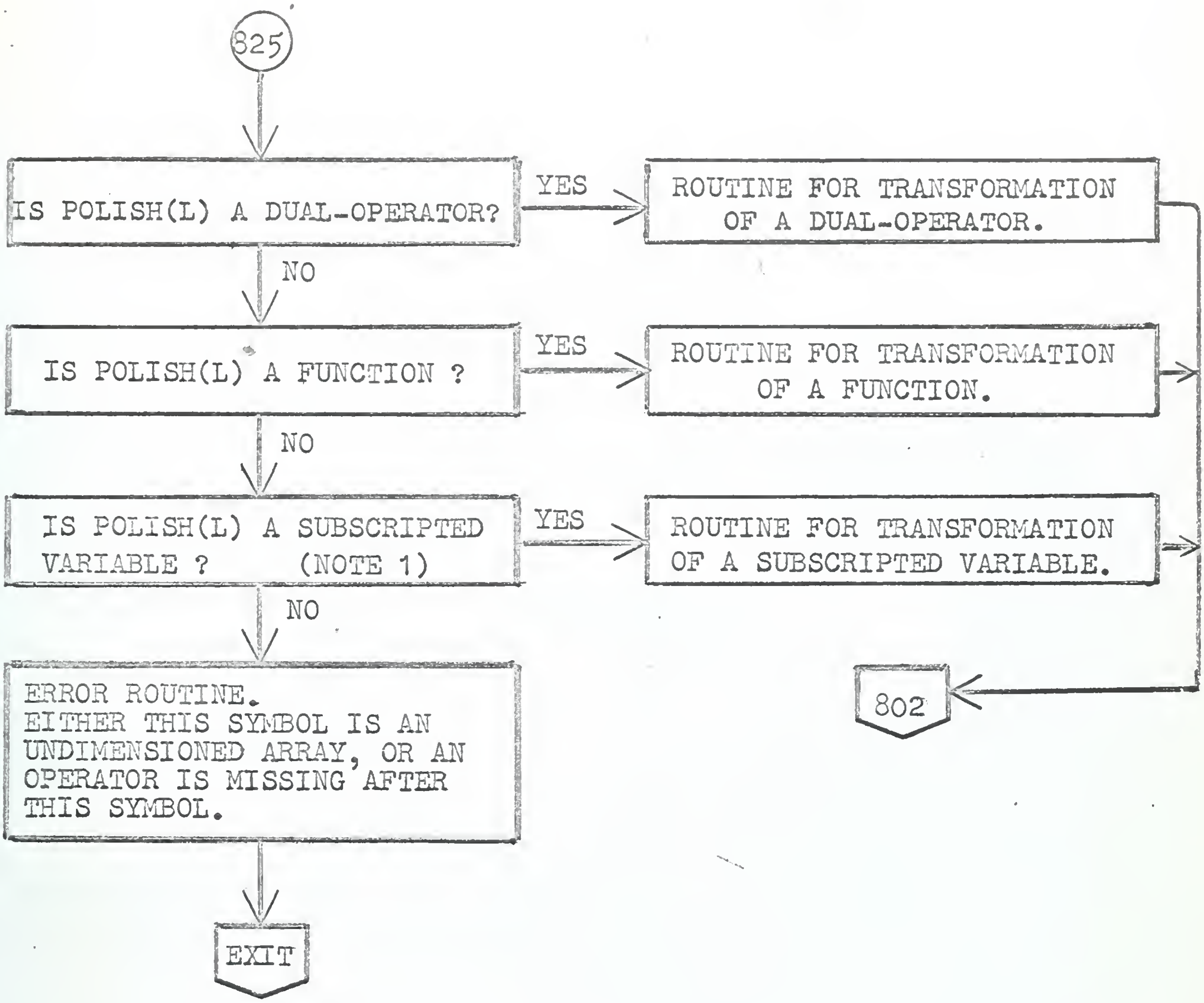
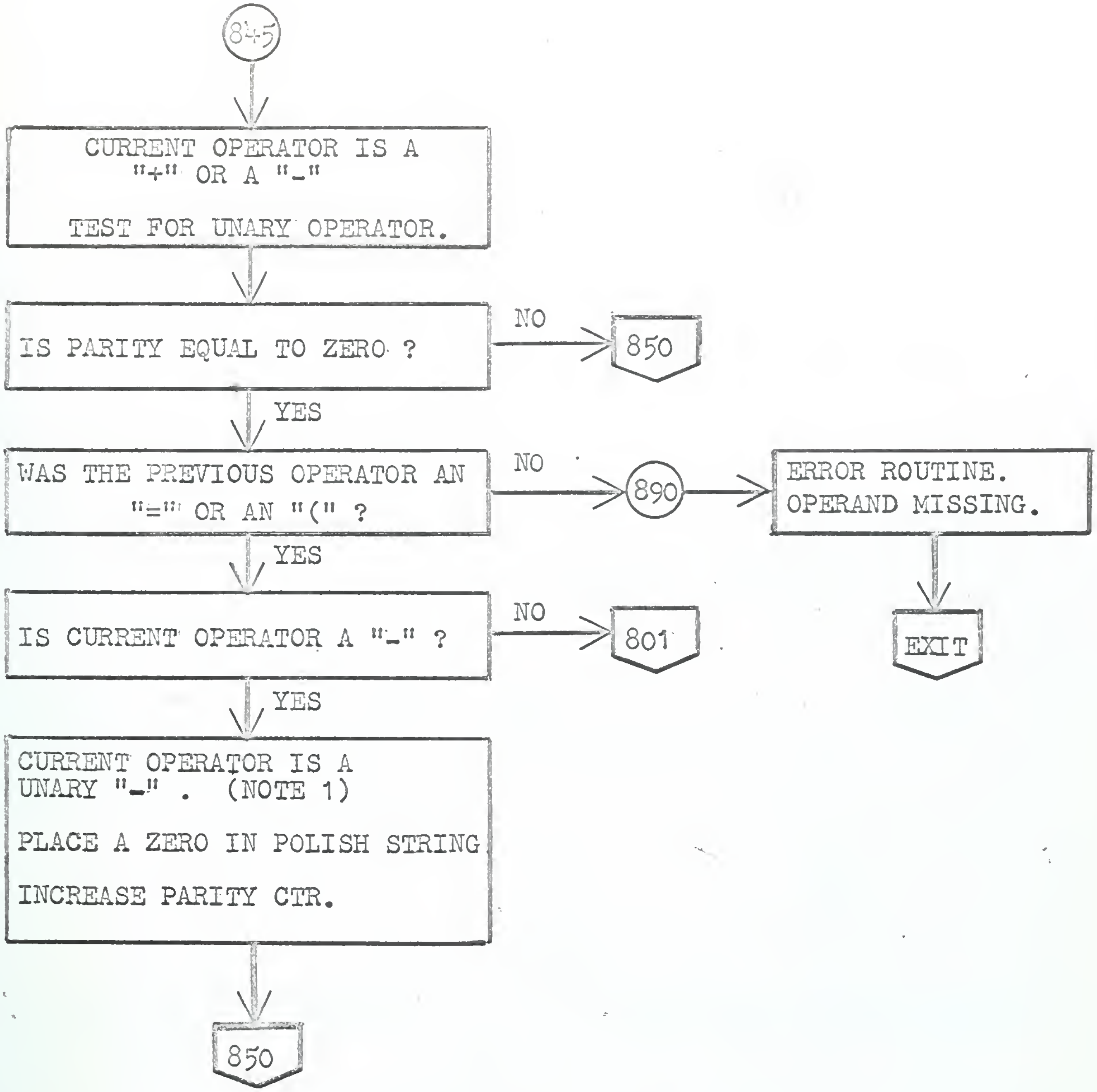


Figure 4 (Cont'd)



NOTE 1. FOR SIMPLICITY, SUBSCRIPTED VARIABLES ARE NOT CONSIDERED HEREIN. THE NECESSARY ROUTINE SHOULD BE INSERTED INTO PROGRAM AT THIS POINT, GIVING CONSIDERATION TO PARITY COUNTERS AND TESTS.

Figure 4 (Cont'd)



NOTE 1. A UNARY "-" IS TREATED AS (0 - OPERAND).

Figure 4 (Cont'd)

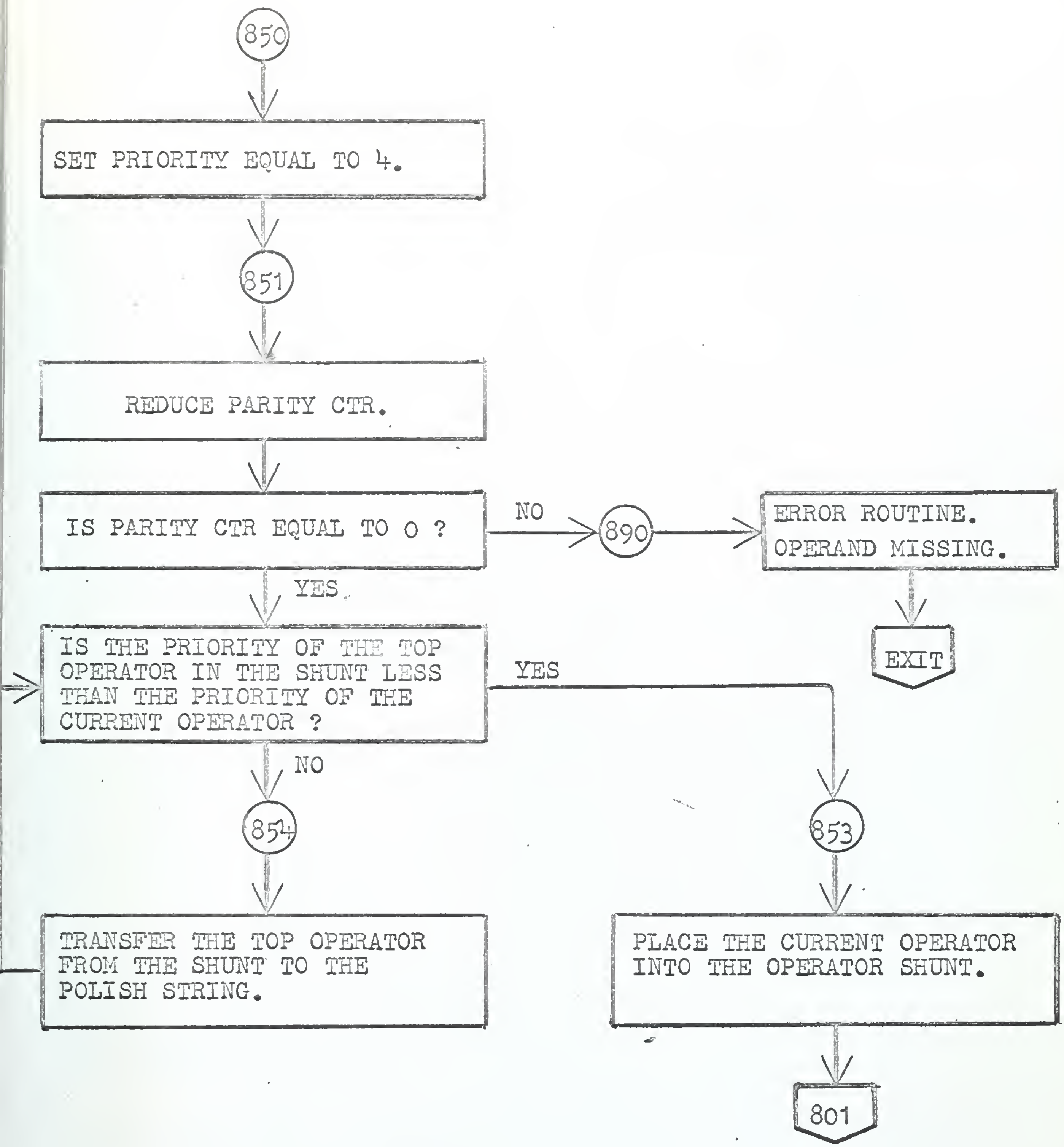


Figure 4 (Cont'd)

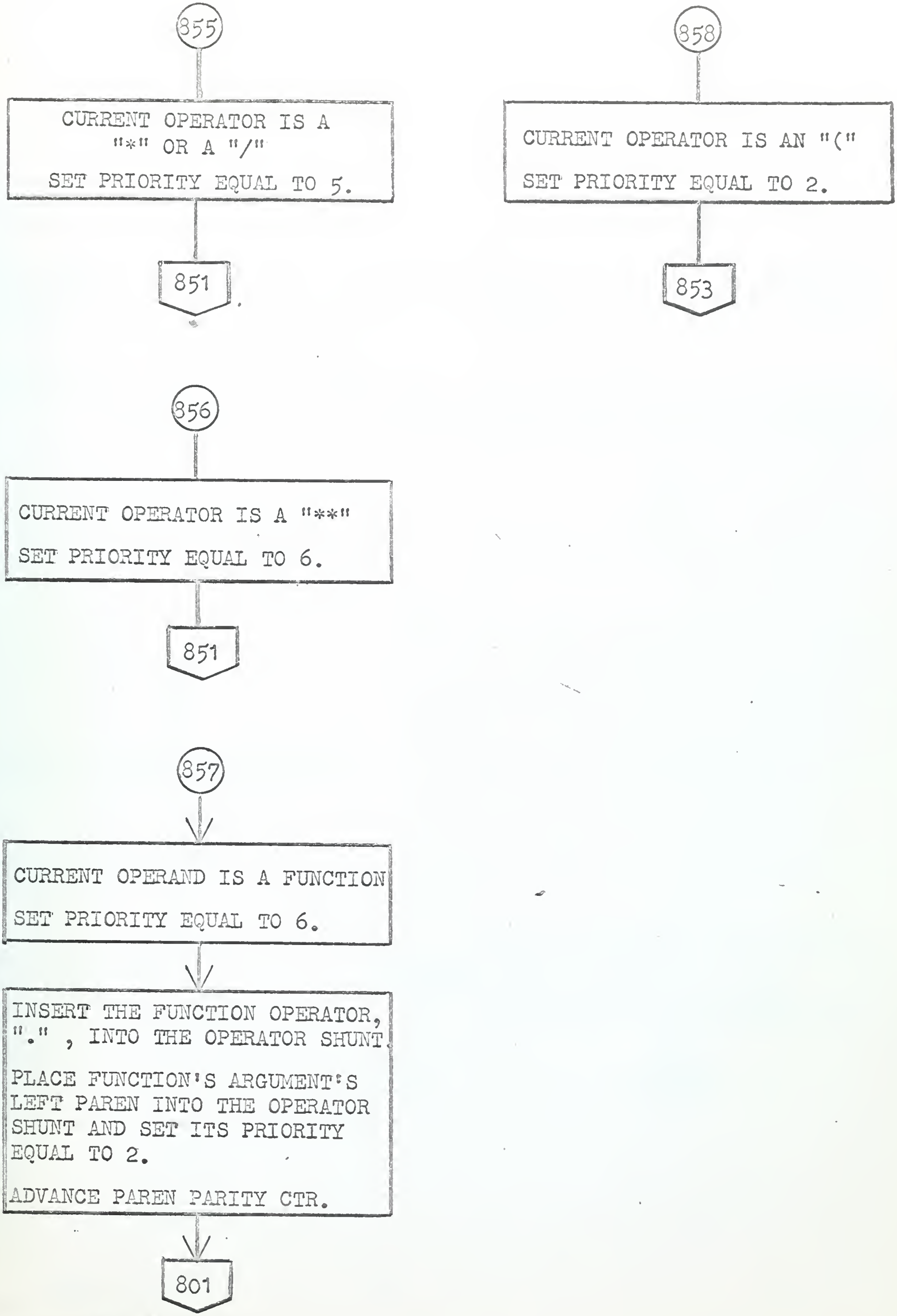


Figure 4 (Cont'd)

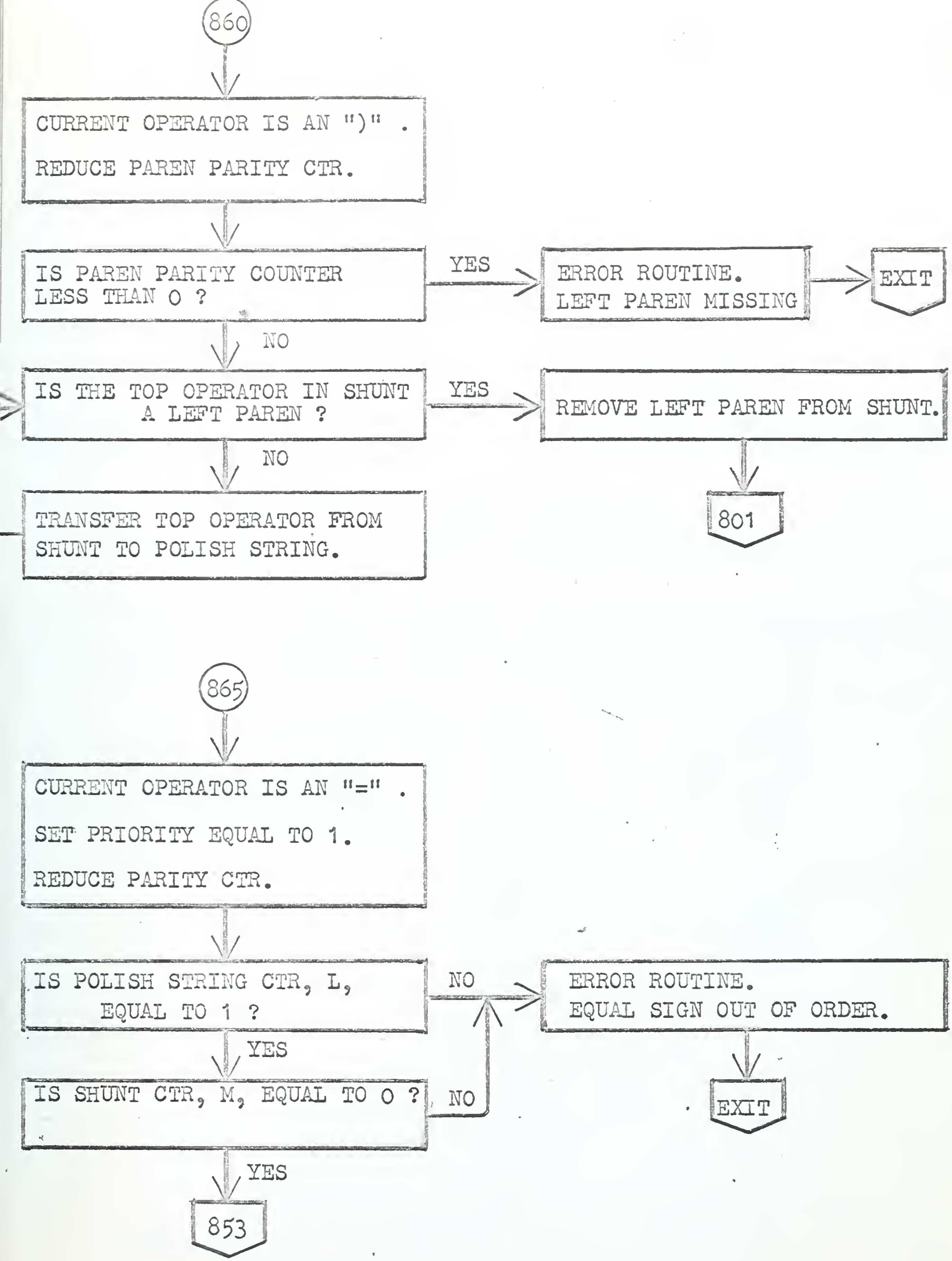


Figure 4 (Cont'd)

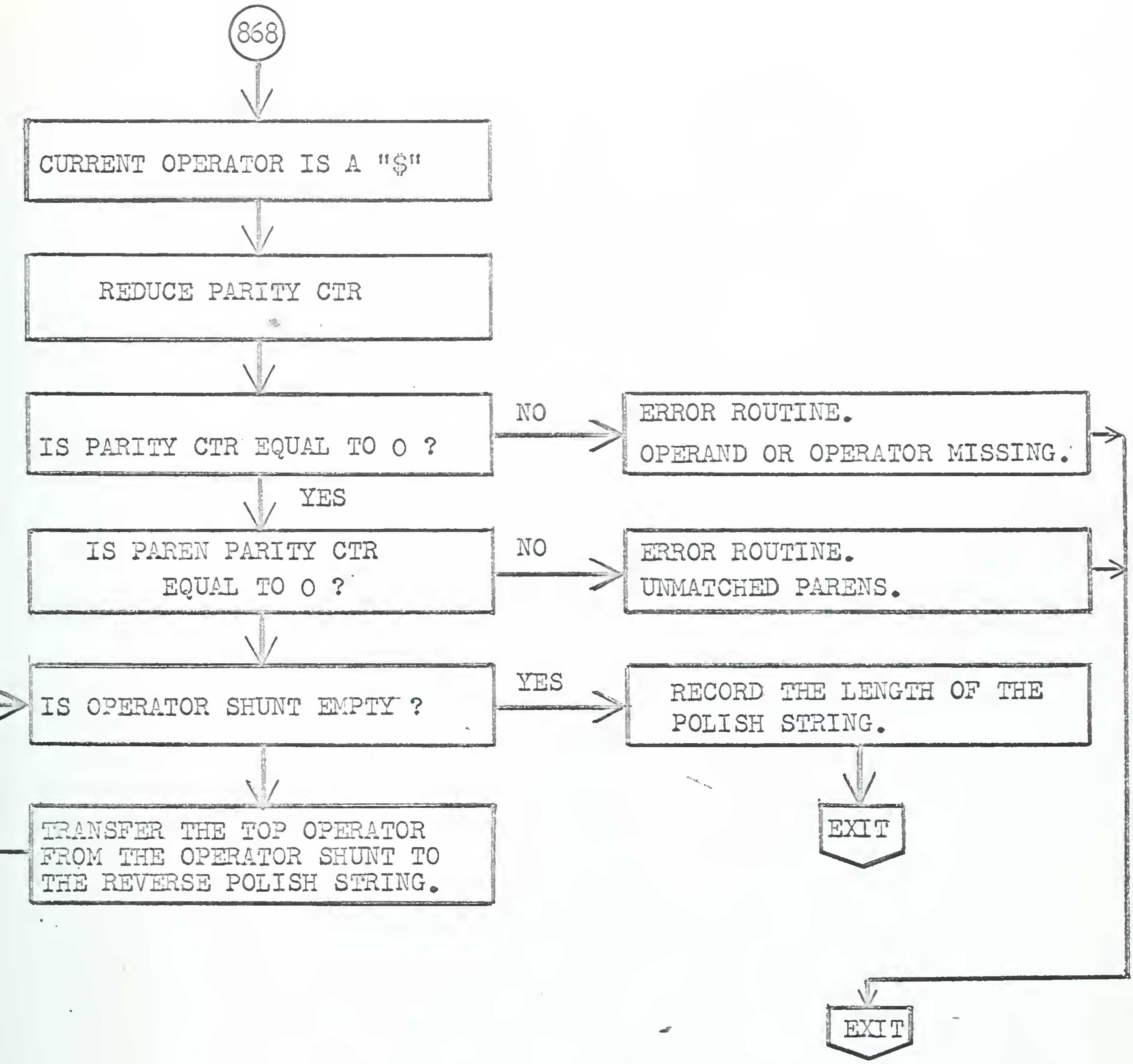
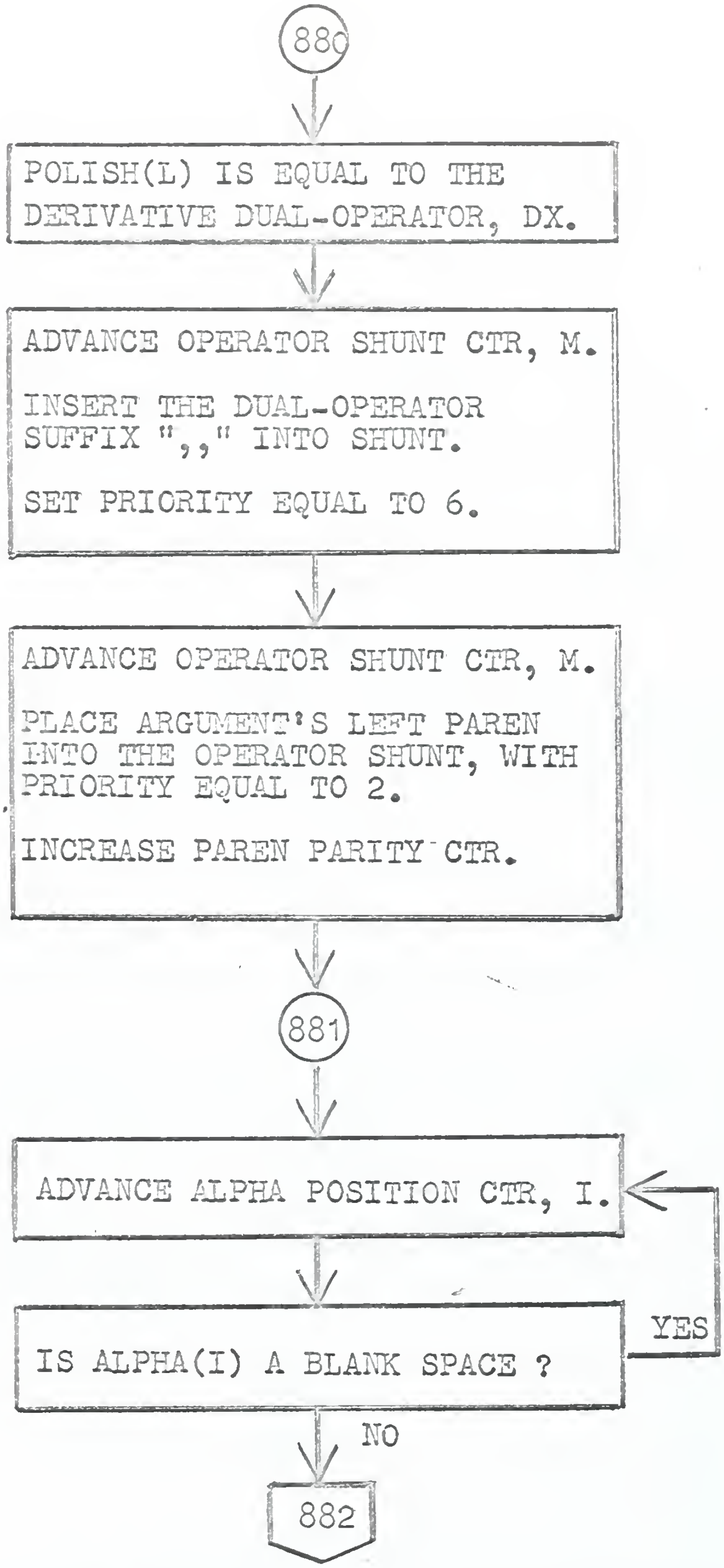


Figure 4 (Cont'd)



NOTE 1. FOR SIMPLICITY, THIS ROUTINE RESTRICTS VARIABLES OF DIFFERENTIATION TO A SINGLE CHARACTER SYMBOL. IF DESIRED, THE ROUTINE PREVIOUSLY DESCRIBED FOR ASSEMBLY OF A MULTIPLE CHARACTER SYMBOL SHOULD BE INSERTED AT JUNCTION 882.

Figure 4 (Cont'd)

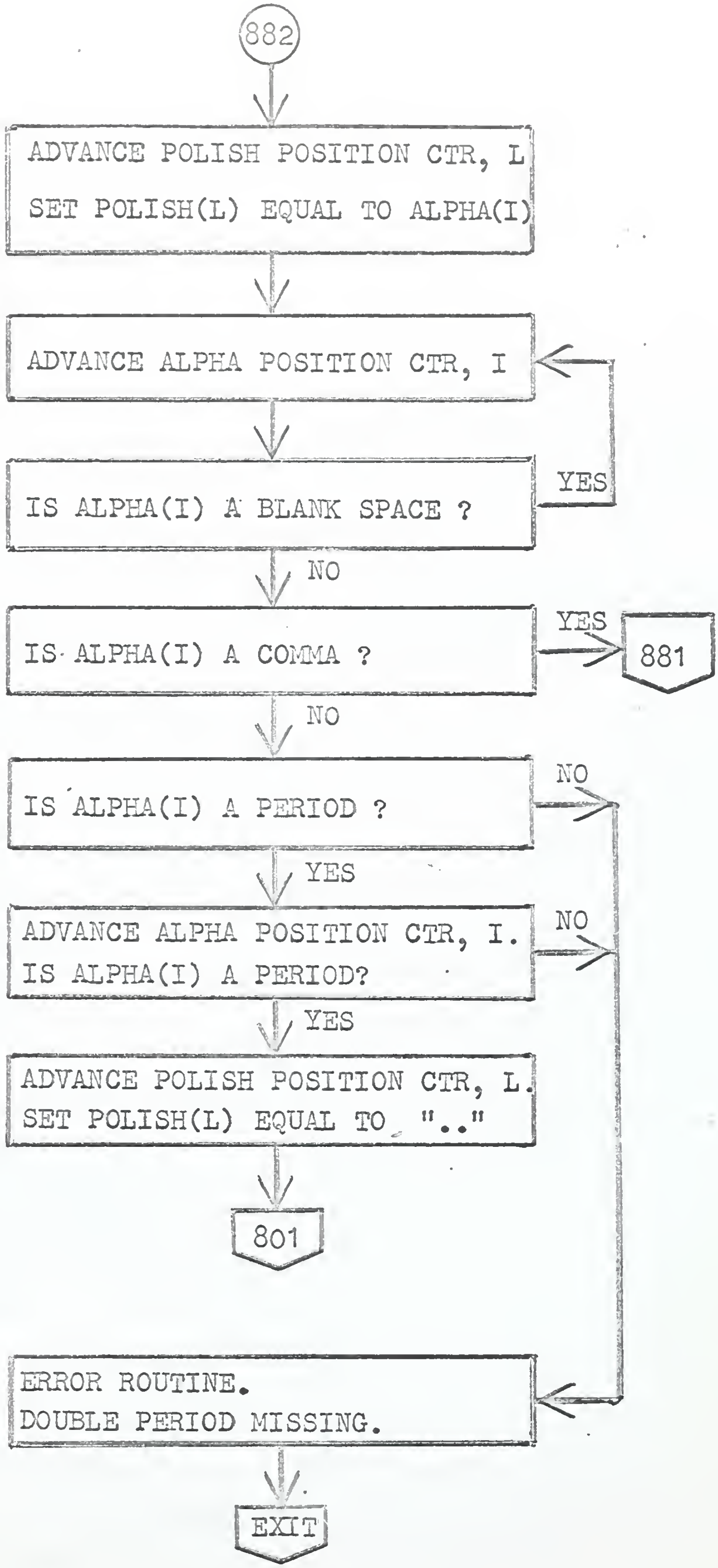


Figure 4 (Cont'd)

DERIVATIVE ROUTINE

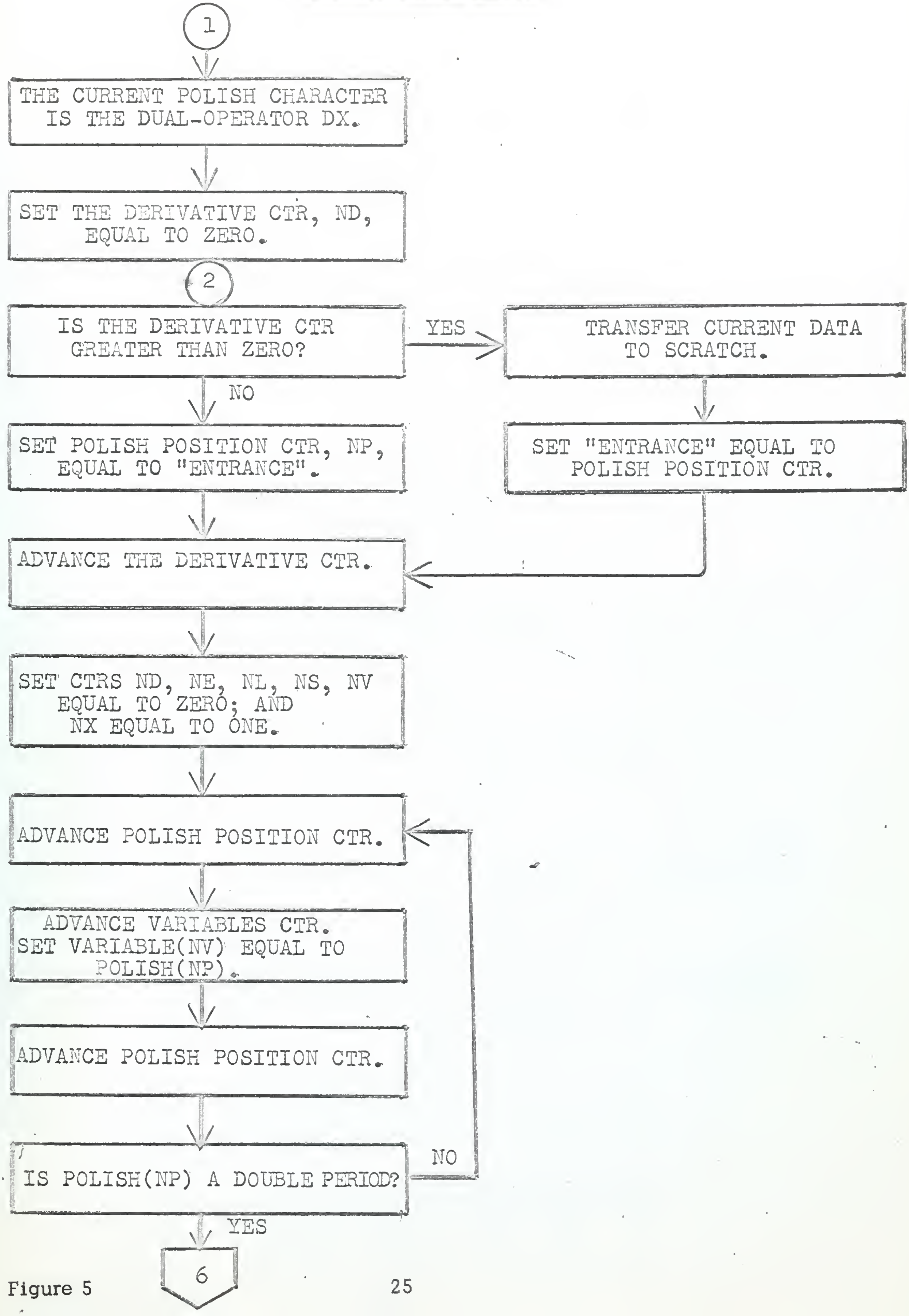


Figure 5

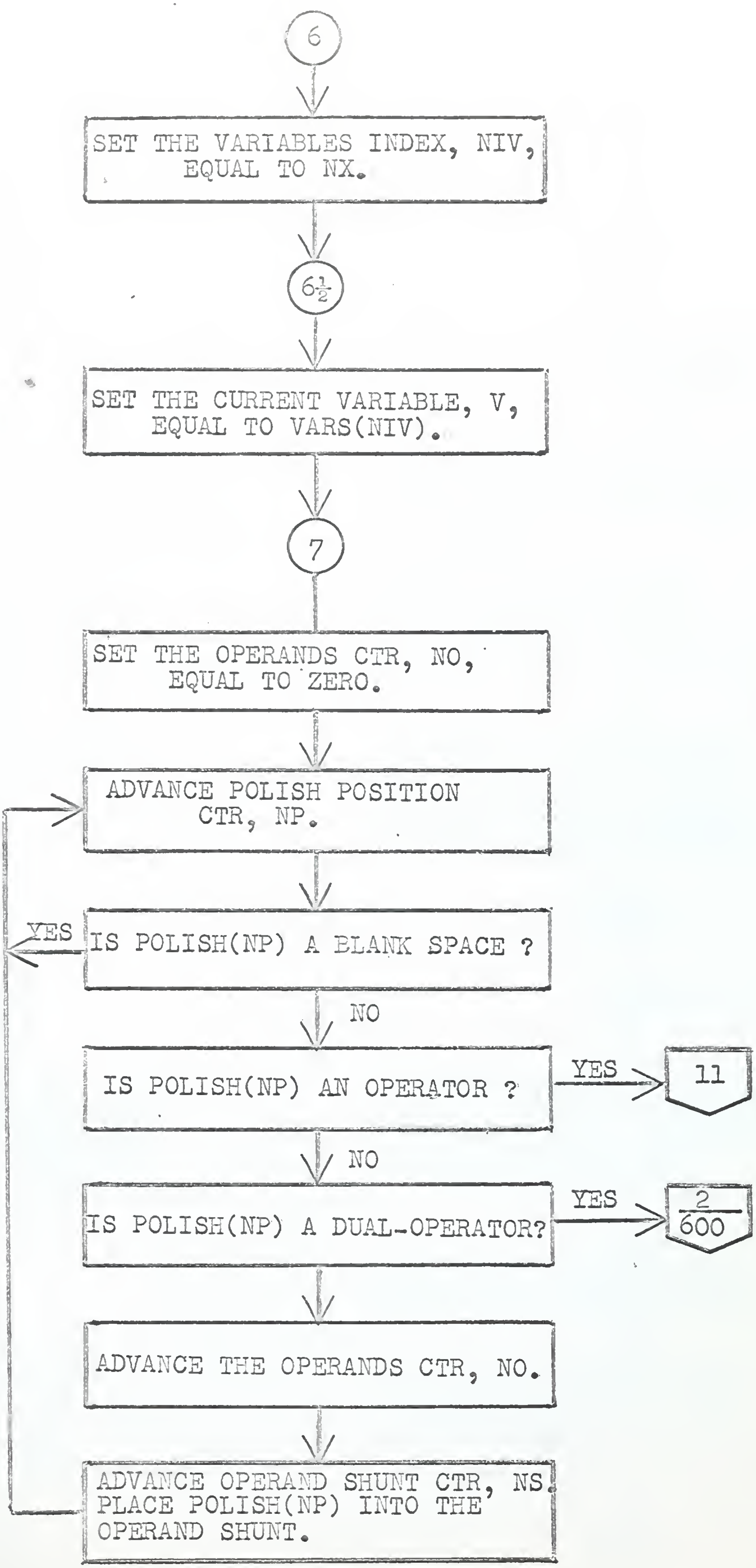


Figure 5 (Cont'd)

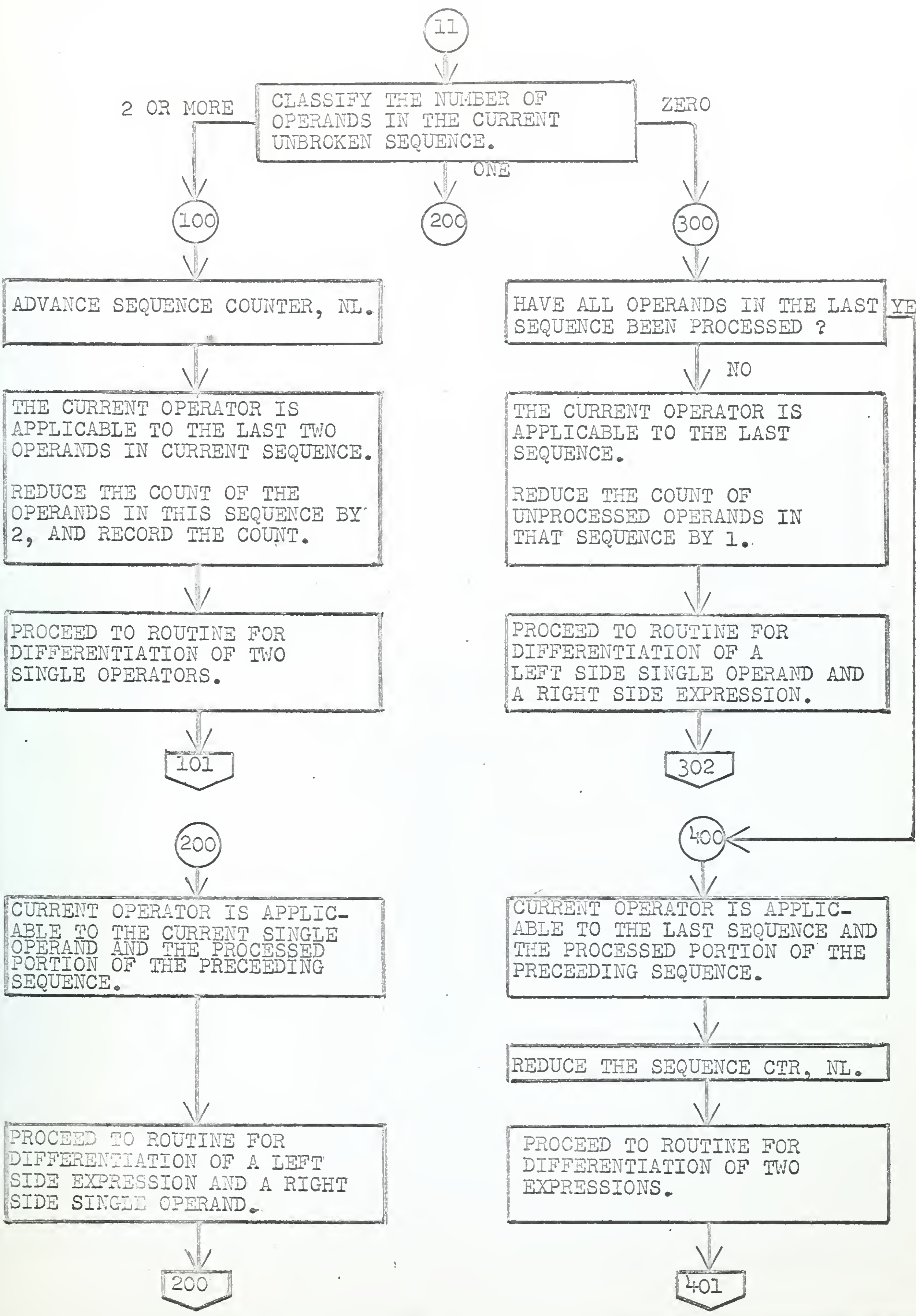


Figure 5 (Cont'd)

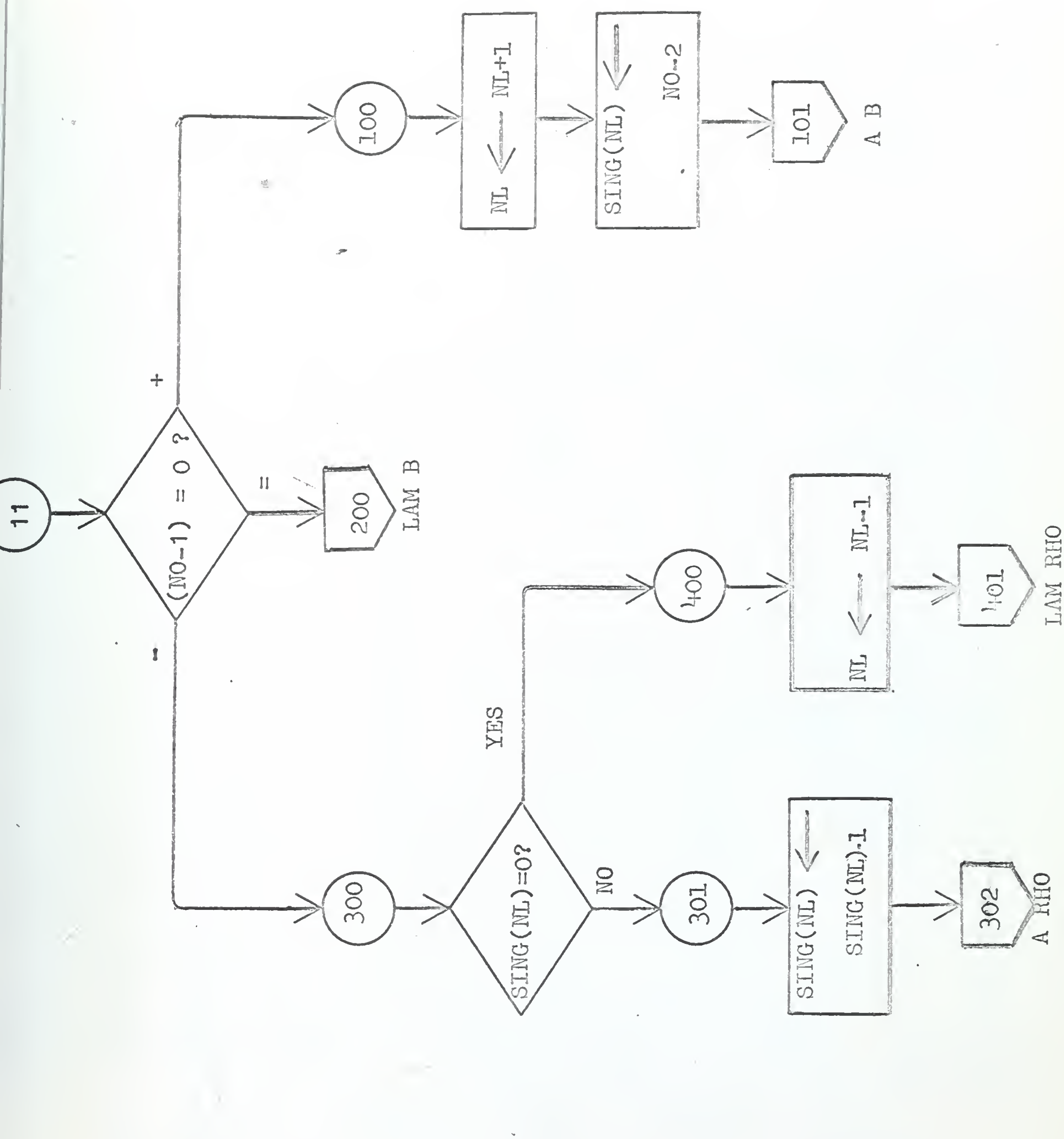


Figure 5 (Cont'd)

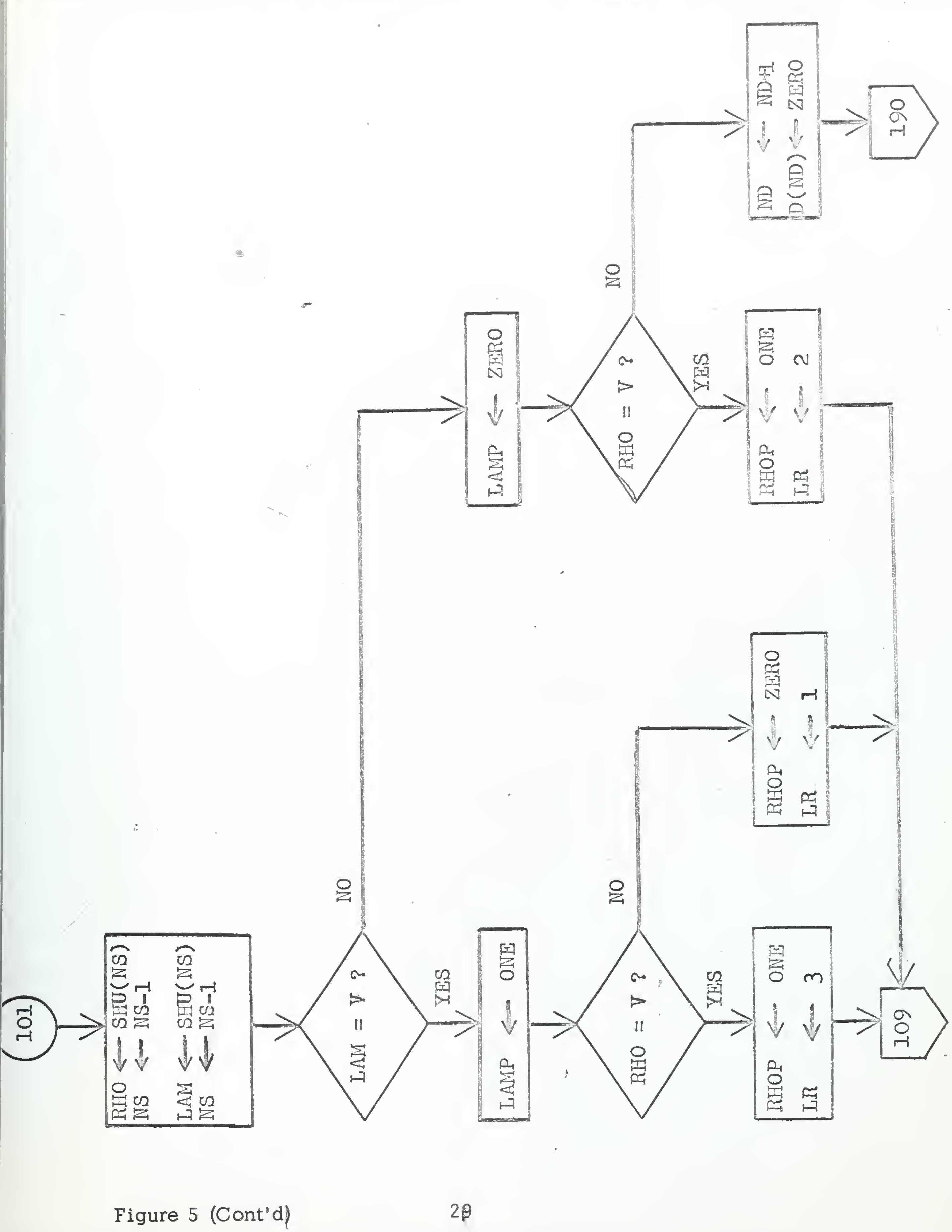


Figure 5 (Cont'd)

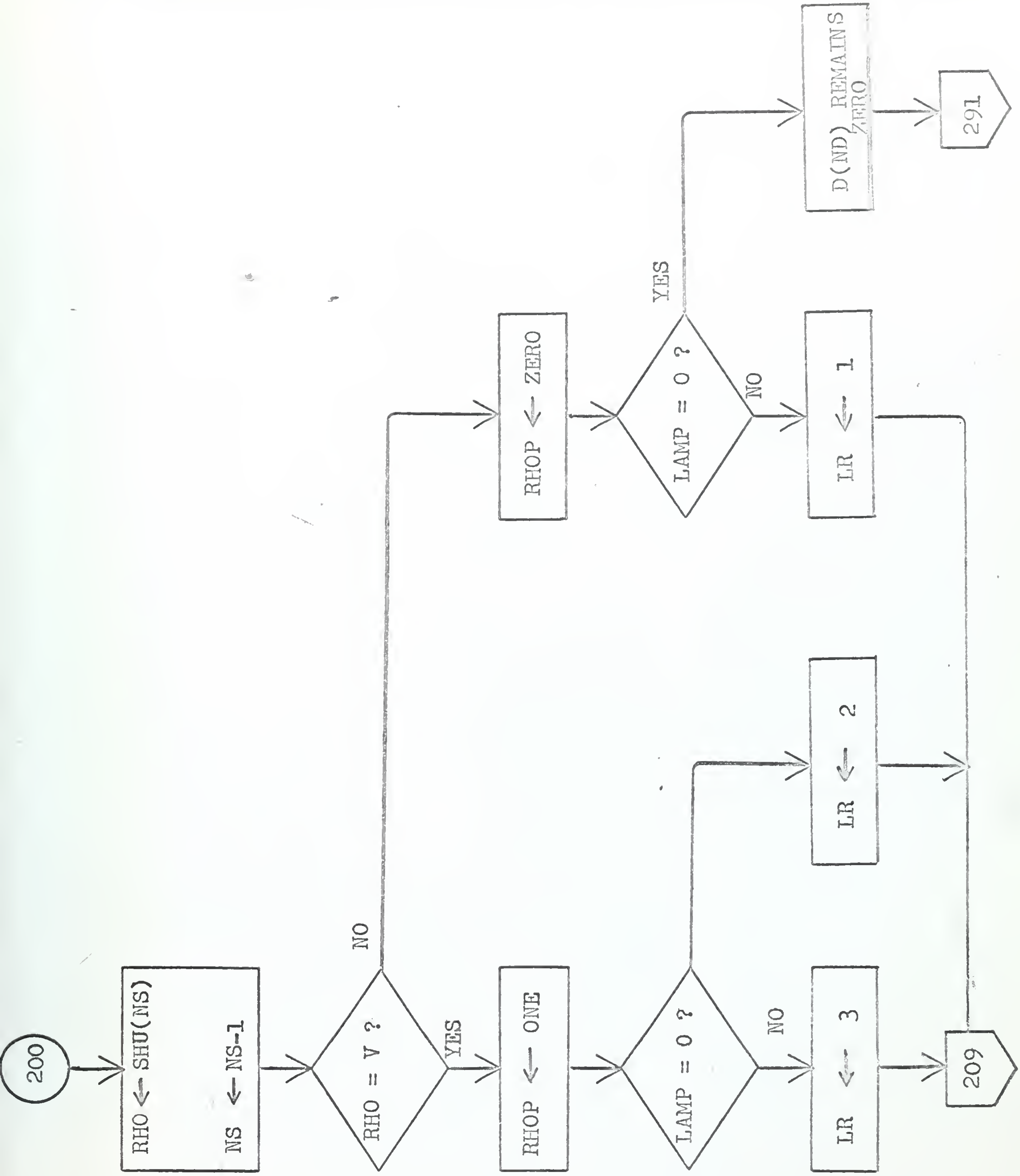


Figure 5 (Cont'd)

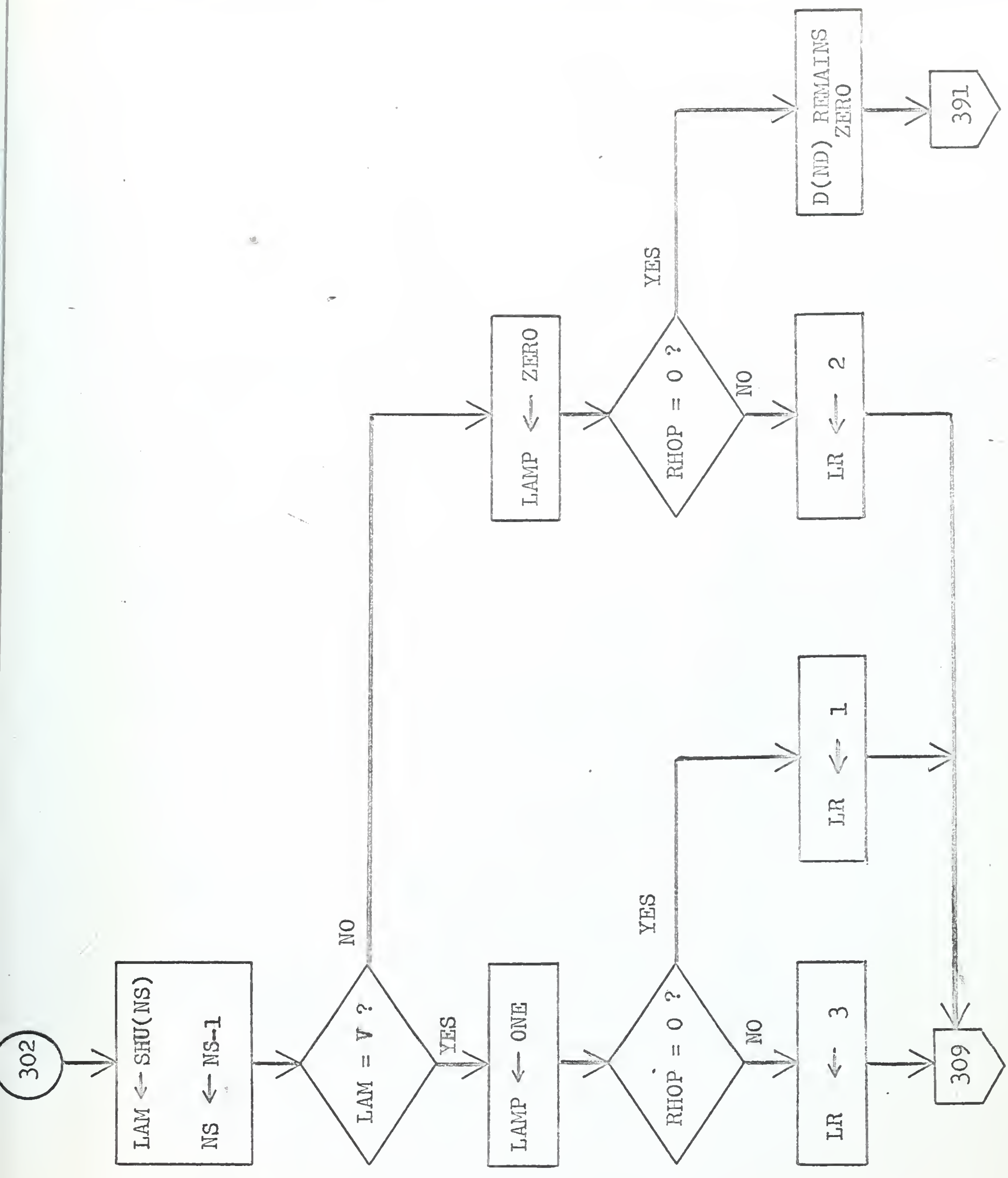


Figure 5 (Cont'd)

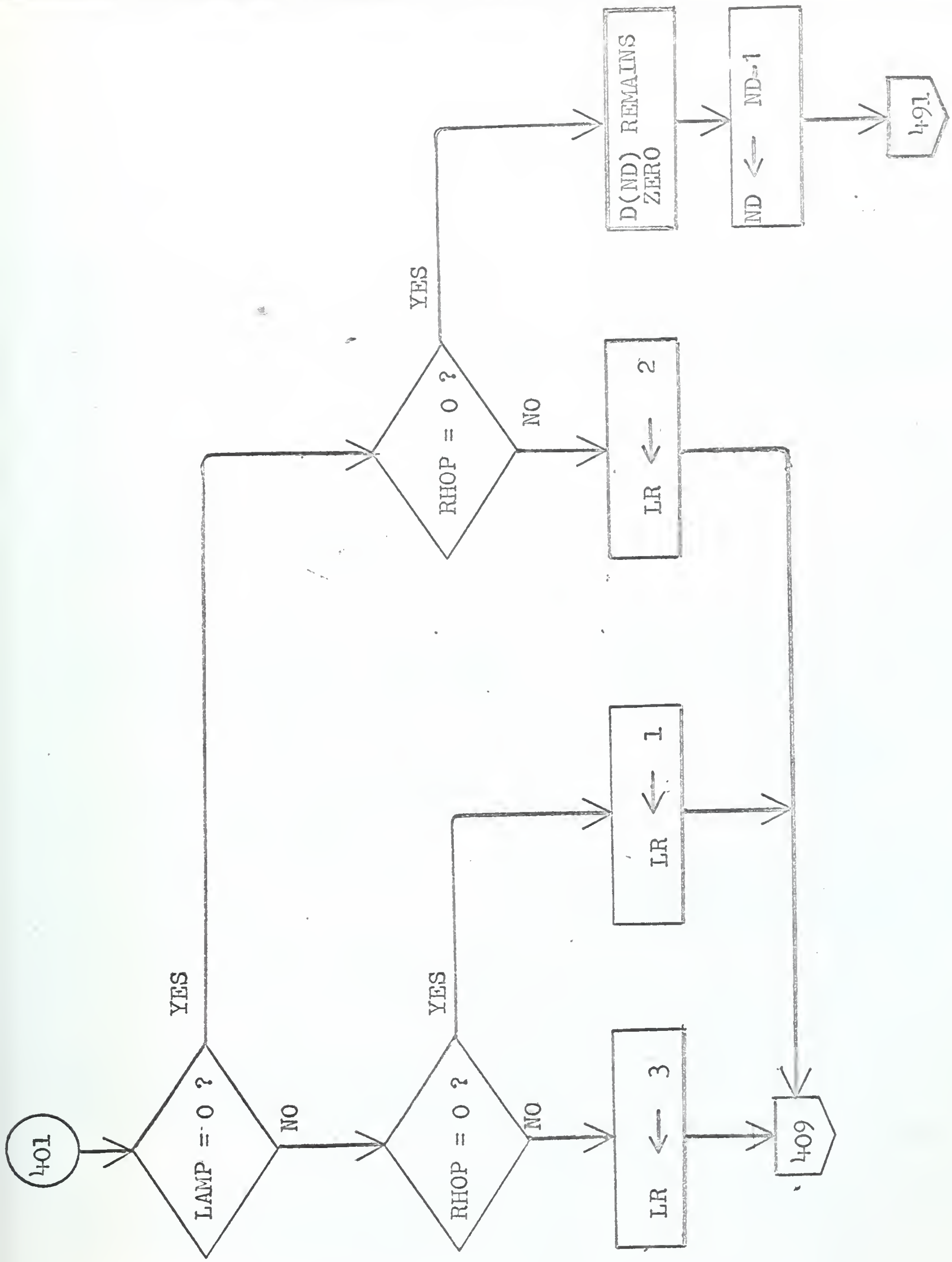


Figure 5 (Cont'd)

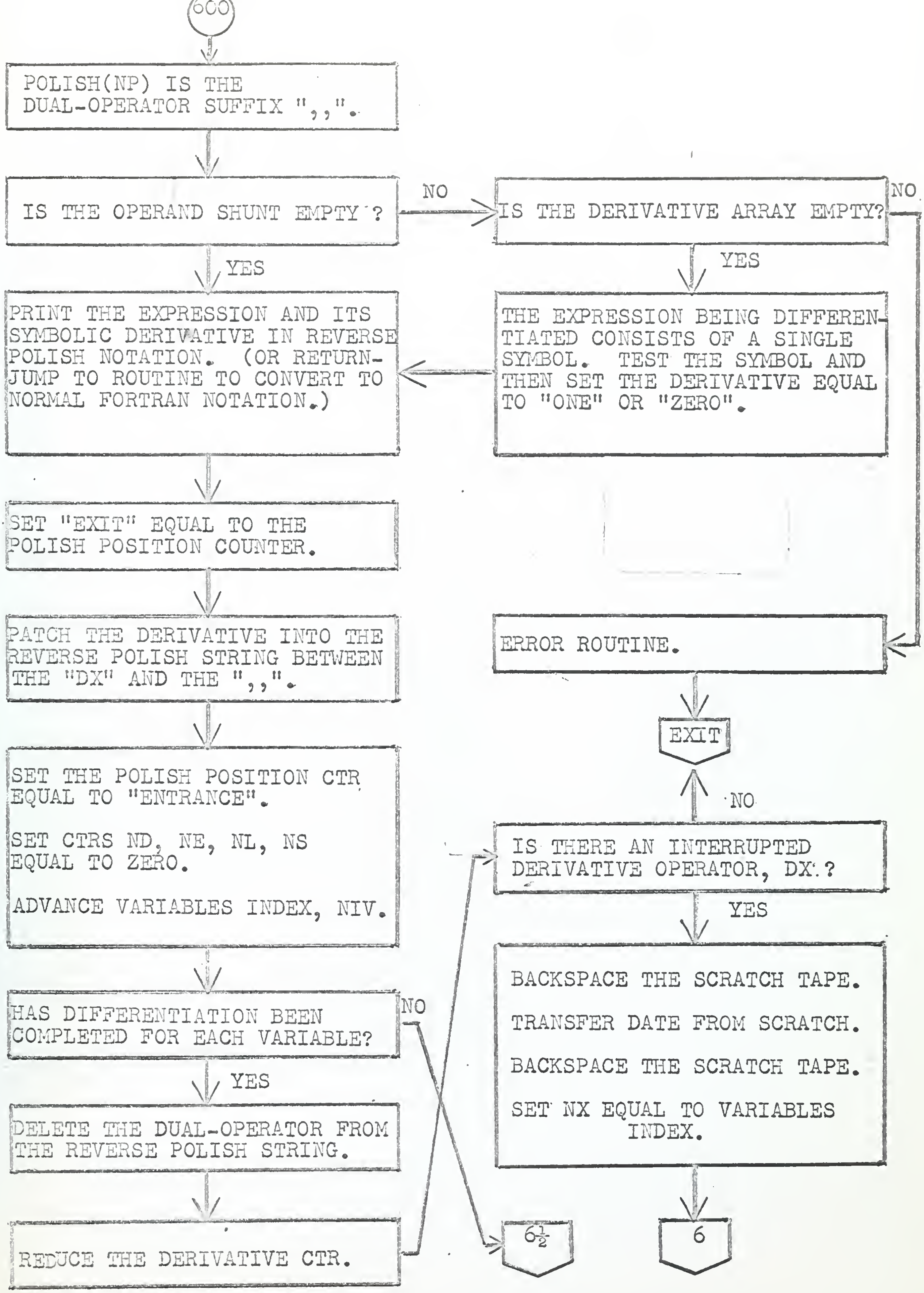


Figure 5 (Cont'd)

```
 TYPE INTEGER  D, E, V, POL, RHO, RHOP, SING, SHU, VARS,
1              BEGIND, BEGINE, ENDD, ENDE, ENTRANCE, EXIT,TERMINUS,
2              DUALOP, OP, FUNCTION,
3              ZERO, ONE, TWO, DIV, PLUS, POWER, FUNC, FIXONE,
4              ALPHA, BUFFER, CHARACTR, PARITY, PRIORITY, Z,
5              PSHUNT, SHUNT

 DIMENSION  DUALOP(2), OP(10), FUNCTION(6), POL(5000),
1           D(3000), E(1000), VARS(50), SHU(100), SING(100),
2           BEGIND(100), BEGINE(100), ENDD(100), ENDE(100),
3           ALPHA(10000), BUFFER(80), PSHUNT(100), SHUNT(100),
4           CHARACTR(10)

 DATA ((OP(N), N=1,10)= 1H+, 1H-, 1H*, 1H/, 2H**, 1H., 1H(, 1H),
1                       1H=, 1H$)
 DATA((FUNCTION(N),N=1,6)= 4HLOGF, 4HEXPF, 4HSINF, 4HCOSF, 4HTANF,
1                          5HSQRTF)
 DATA ((DUALOP(N), N=1,2)=  2HDX,  2H,,)

 ZERO=2H0. $ ONE=2H1. $ TWO=2H2. $
 PLUS=1H+ $ MINUS=1H- $ MUL=1H* $ DIV=1H/ $ POWER = 2H** $
 FUNC = 1H. $ FIXONE = 1H1
```

Figure 6

```
C          ROUTINE TO CONTROL TEST

10001 READ 10002,  (BUFFER(J), J=1,72)
10002 FORMAT (72A1)

C DATA CARDS MUST CONTAIN ONE CARD (EQUATION) MORE THAN SPECIFIED IN DO LOOP

      DO 999 NTEST=1,30

10004 DO 10005 I=1,72
10005 ALPHA(I) = BUFFER(I)
      I=72
10006 READ 10002, (BUFFER(J),J=1,72)
      IF (BUFFER(6) .EQ. 1H ) 10009, 10007
10007 DO 10008 J=7,72
      I=I+1
10008 ALPHA(I)=BUFFER(J)
      GO TO 10006
10009 I=I+1
      ALPHA(I)=1H$
```

Figure 6 (Cont'd)

```
C          ENTRY TO ROUTINE TO TRANSFORM FORTRAN 63 ALGEBRAIC EQUATION
C          INTO REVERSE POLISH NOTATION.

C          INITIALIZE COUNTERS
  800 I=6 $ J=0 $ K=0 $ L=0 $ M=0 $ PARITY=0 $ Z=0
  801 I=I+1
  802 IF(ALPHA(I) .EQ. 1H ) 801,803
  803 DO 810 N=1,10
  804 IF(ALPHA(I) . EQ. OP(N)) 805,810
  805 IF(ALPHA(I) .EQ. 1H*) 806,808
  806 I=I+1
      IF(ALPHA(I) .EQ. 1H*) 807, 809
C 807     OPERATOR IS A **
  807 ALPHA(I)=2H**
      GO TO 856
  808 GO TO (845,845,855,855,856,811,858,860,865,868) N
  809 I=I-1
      GO TO 808
  810 CONTINUE
  811 K=1
      CHARACTR(K)=ALPHA(I)
  812 I=I+1
      IF(ALPHA(I) .EQ. 1H ) 820, 813
  813 DO 814 N=1,5
      IF(ALPHA(I) .EQ. OP(N)) 820,814
  814 CONTINUE
      DO 815 N=7,10
      IF (ALPHA(I) .EQ. OP(N)) 820, 815
  815 CONTINUE
      K=K+1
      CHARACTR(K)=ALPHA(I)
      IF(K .GT. 8) 816,812
  816 PRINT 817,(CHARACTR(K), K=1,8)
  817 FORMAT (//31H ERROR. SYMBOL BEGINNING WITH (, 8A1,  23H) EXCEEDS 8
     1 CHARACTERS.)
      GO TO 999

  820 I=I-1
      L=L+1
      ENCODE (8, 821, POL(L))  (CHARACTR(J), J=1,K)
  821 FORMAT (8A1)
```

Figure 6 (Cont'd)

```
  822 I=I+1
      IF(ALPHA(I) .EQ. 1H ) 822,823
  823 IF(ALPHA(I) .EQ. 1H() 825,824
C 824      POL(L) IS A SYMBOL
  824 PARITY=PARITY+1
      IF(PARITY .EQ. 1) 802, 892

  825 IF (POL(L) .EQ. 2HDX) 880, 826

  826 DO 828 N=1, 6
  827 IF (POL(L) .EQ. FUNCTION(N)) 857, 828
  828 CONTINUE
      PRINT 829, POL(L)
  829 FORMAT (//28H ERROR.  EITHER THE SYMBOL (,  A8,  79H) IS AN UNDIME
     CNSIONED ARRAY OR AN OPERATOR SHOULD BE INSERTED AFTER THE SYMBOL.)
      GO TO 999

C          SUBSCRIPTED VARIABLES ARE NOT ALLOWED IN THIS PROGRAM.
C          A ROUTINE TO DETERMINE IF POL(L) IS A SUBSCRIPTED VARIABLE
C          CAN BE INSERTED AT THIS POINT.

  842 PRINT 843
  843 FORMAT (//81H IMPROPER EXPRESSION.  DOUBLE PERIOD MISSING AFTER
     1VARIABLE OF DIFFERENTIATION.)
      GO TO 999

C 845      TEST FOR UNARY + OR -
  845 IF (PARITY .EQ. 0) 846, 850
C 846     LAST CHARACTER WAS AN OPERATOR.
  846 IF (SHUNT(M) .EQ. 1H=  .OR. SHUNT(M) .EQ. 1H( ) 847, 890
C 847      UNARY + OR -
  847 IF (N .EQ. 1) 801, 848
C 848      UNARY -
  848 L=L+1
      POL(L)=ZERO
      PARITY=PARITY+1
      GO TO 850

C 850      N IS EQUAL TO 1 OR 2.  THE OPERATOR IS A + OR -. PRIORITY IS 4.
  850 PRIORITY = 4
  851 PARITY = PARITY - 1
      IF(PARITY) 890, 852, 892
```

Figure 6 (Cont'd)

```
  852 IF(PSHUNT(M) .LT. PRIORITY) 853, 854
C 853     PLACE CURRENT OPERATOR INTO SHUNT.
  853 M=M + 1
      SHUNT(M) = ALPHA(I)
      PSHUNT(M) = PRIORITY
      GO TO 801
C 854     TRANSFER TOP OPERATOR FROM SHUNT TO REVERSE POLISH STRING.
  854 L=L+1
      POL(L) = SHUNT(M)
      M = M - 1
      GO TO 852

C 855     N IS EQUAL TO 3 OR 4.  THE OPERATOR IS A * OR /.  PRIORITY IS 5.
  855 PRIORITY = 5
      GO TO 851

C 856     N IS EQUAL TO 5.  THE OPERATOR IS A  **.  ITS PRIORITY IS 6.
  856 PRIORITY = 6
      GO TO 851

C 857     FUNCTION ROUTINE.  MUST INSERT THE OPERATOR (.)  WHOSE PRIORITY IS 7.
  857 M=M+1
      PSHUNT(M) = 7
C         ALPHA(I) CONTAINS FUNCTIONS ARGUMENTS OPEN PAREN,  WHICH MUST
      SHUNT(M) = 1H.
C            NOW BE PLACED INTO THE SHUNT.
      M=M+1
      SHUNT(M) = ALPHA(I)
      PSHUNT(M)=2
      Z=Z+1
      GO TO 801

C 858     N IS EQUAL TO 7.  THE OPERATOR IS A (.  ITS PRIORITY IS 2.
  858 PRIORITY = 2
C         ADVANCE PAREN PARITY COUNTER.
      Z = Z + 1
      GO TO 853

C 860     N IS EQUAL TO 8.  THE OPERATOR IS A ).  ITS PRIORITY IS 3.
C 860
  860 IF(PSHUNT(M) - 2) 894,861,863
  861 Z = Z - 1
```

Figure 6 (Cont'd)

```
      IF(Z .LT. 0) 894, 862
C 862     REMOVE LEFT PAREN FROM SHUNT.
  862 M = M - 1
      GO TO 801
C 863     TRANSFER TOP OPERATOR FROM SHUNT TO REVERSE POLISH STRING.
  863 L = L + 1
      POL(L) = SHUNT(M)
      M = M - 1
      GO TO 860

C 865     N IS EQUAL TO 9.  THE OPERATOR IS A =.  ITS PRIORITY IS 1.
  865 PRIORITY = 1
      PARITY = PARITY - 1
      IF(M .EQ. 0) 866, 898
  866 IF(L .EQ. 1) 853, 896

C 868     N IS EQUAL TO 10. THE OPERATOR IS A $.  ITS PRIORITY IS 0.
  868 PARITY = PARITY - 1
      IF(PARITY) 890, 869, 892
  869 IF(Z .EQ. 0) 870,894
  870 IF(M .EQ. 0) 871, 875
  871 TERMINUS = L
      PRINT 872, (ALPHA(II) , II=1,I)
  872 FORMAT (//20H           FORTRAN.   , 72A1, (//26X, 66A1))
  873 PRINT 874, (POL(LL), LL=1,L)
  874 FORMAT (//20H           POLISH.    , 6X,  15A6,  (//26X, 15A6))
C         TRANSFORMATION INTO REVERSE POLISH NOTATION COMPLETED.   CONTINUE.
      GO TO 900

C 875     TRANSFER ALL OPERATORS REMAINING IN SHUNT TO REVERSE POLISH STRING.
  875 L = L + 1
      POL(L) = SHUNT(M)
      M = M - 1
      GO TO 870

C 880     POLISH(L) IS THE DUAL-OPERATOR DX
  880 M=M+1
      SHUNT(M) = 2H,,
      PSHUNT(M) = 99
      M = M + 1

C         ALPHA(I) CONTAINS A LEFT PAREN. PLACE IT INTO SHUNT.
```

Figure 6 (Cont'd)

```
      SHUNT(M) = ALPHA(I)
      PSHUNT(M)=2
      Z=Z+1
  881 I = I + 1
      IF(ALPHA(I) .EQ. 1H ) 881, 882

C 882      ALPHA(I) CONTAINS VARIABLE OF DIFFERENTIATION. PLACE IT INTO POLISH
  882 L = L + 1
      POL(L) = ALPHA(I)
  883 I = I + 1
      IF(ALPHA(I) .EQ. 1H ) 883, 884
  884 IF(ALPHA(I) .EQ. 1H,) 881, 886
  886 IF(ALPHA(I) .EQ. 1H.) 887, 842
  887 I = I + 1
      IF(ALPHA(I) .EQ. 1H.) 888, 842
  888 L = L + 1
      POL(L) = 2H..
      GO TO 801

  890 PRINT 891
  891 FORMAT (/41H IMPROPER EXPRESSION. IDENTIFIER MISSING.)
      GO TO 999
  892 PRINT 893
  893 FORMAT (/39H IMPROPER EXPRESSION. OPERATOR MISSING.)
      GO TO 999
  894 PRINT 895
  895 FORMAT (/51H IMPROPER EXPRESSION. LEFT/RIGHT PARENS UN-MATCHED.)
      GO TO 999
  896 PRINT 897
  897 FORMAT (/53H INPROPER EXPRESSION. EQUAL SIGN OUT OF PROPER ORDER.)
      GO TO 999
  898 PRINT 899
  899 FORMAT (/65H IMPROPER EXPRESSION. LEFT SIDE IDENTIFIER MISSING FRO
     CM EQUATION.)
      GO TO 999

  900 CONTINUE
```

Figure 6 (Cont'd)

```
C          ENTRY TO ROUTINE TO TRANSFORM REVERSE POLISH NOTATION INTO
C          CDC 1604 MACHINE SYMBOLIC MACROS.


      DO 1001 I=1,TERMINUS
      IF(POL   (I) .EQ. 2HDX)55555,2300
55555 MP=I
      ENTRANCE =I
      GO TO 1
C2300     CHECK TO SEE IF OPERATOR IS CONTAINED IN POLISH STRING
 2300 DO  3000 N=1,9
      IF(POL(I) .EQ. OP(N))3100,3000
 3100 GO TO (1200,1200,1200,1200,4700,1400,3000,3000,1100)N
 3000 CONTINUE
C        CHECK TO SEE IF FUNCTION IS CONTAINED IN POLISH STRING
      DO 4000 LN=1,7
      IF(POL(I) .EQ. FUNCTION(LN))5000,4000
 5000 GO TO 1500
 4000 CONTINUE
      GO TO 2700
4700  IF(POL   (I-1) .EQ. 1H1)4800,5400
5400  IF(POL   (I-1) .EQ. 1H2)4900,5500
5500  IF(POL   (I-1) .EQ. 1H3)5100,5600
5600  IF(POL   (I-1) .EQ. 1H4)5200,5700
5700  IF(POL   (I-1) .EQ. 1H5)1111,1112
1112  IF(POL   (I-1) .EQ. 1H6)1113,1114
1114  IF(POL   (I-1) .EQ. 1H7)1115,1116
1116  IF(POL   (I-1) .EQ. 1H8)1117,1118
1118  IF(POL   (I-1) .EQ. 1H9)1119,1120
1120  IF(POL   (I-1) .EQ. 2H10)1121,1122
1122  IF(POL   (I-1) .EQ. 2H11)1123,1124
1124  IF(POL   (I-1) .EQ. 2H12)1125,1126
1126  IF(POL   (I-1) .EQ. 2H13)1127,1128
1128  IF(POL   (I-1) .EQ. 2H14)1129,1130
1130  IF(POL   (I-1) .EQ. 2H15)1131,1132
1132  IF(POL   (I-1) .EQ. 2H16)1133,1134
1134  IF(POL   (I-1) .EQ. 2H17)1135,1136
1136  IF(POL   (I-1) .EQ. 2H18)1137,1138
1138  IF(POL   (I-1) .EQ. 2H19)1139,1140
1140  IF(POL   (I-1) .EQ. 2H20)1141,1142
1142  IF(POL   (I-1) .EQ. 2H21)1143,1144
1144  IF(POL   (I-1) .EQ. 2H22)1145,1146
```

Figure 6 (Cont'd)

```
1146  IF(POL   (I-1) .EQ. 2H23)1147,1148
1148  IF(POL   (I-1) .EQ. 2H24)1149,1150
1150  IF(POL   (I-1) .EQ. 2H25)1151,1152
1152  IF(POL   (I-1) .EQ. 2H26)1153,1154
1154  IF(POL(I-1) .EQ. 2H27)1155,1156
1156  CALL SUB5(I)
      GO TO 1001
2700  IF(POL   (I) .EQ. 1H )1001,2800
2800  CALL MICRO(POL    ,I)
      GO TO 1001
4800  NUM=1
      GO TO 1300
4900  NUM=2
      GO TO 1300
5100  NUM=3
      GO TO 1300
5200  NUM=4
      GO TO 1300
1111  NUM=5
      GO TO 1300
1113  NUM=6
      GO TO 1300
1115  NUM=7
      GO TO 1300
1117  NUM=8
      GO TO 1300
1119  NUM=9
      GO TO 1300
1121  NUM=10
      GO TO 1300
1123  NUM=11
      GO TO 1300
1125  NUM=12
      GO TO 1300
1127  NUM=13
      GO TO 1300
1129  NUM=14
      GO TO 1300
1131  NUM=15
      GO TO 1300
1133  NUM=16
      GO TO 1300
```

Figure 6 (Cont'd)

```
1135  NUM=17
      GO TO 1300
1137  NUM=18
      GO TO 1300
1139  NUM=19
      GO TO 1300
1141  NUM=20
      GO TO 1300
1143  NUM=21
      GO TO 1300
1145  NUM=22
      GO TO 1300
1147  NUM=23
      GO TO 1300
1149  NUM=24
      GO TO 1300
1151  NUM=25
      GO TO 1300
1153  NUM=26
      GO TO 1300
1155  NUM=27
      GO TO 1300
1200  CALL SUB1(POL    ,I)
      GO TO 1001
1300  CALL SUB2(NUM)
      GO TO 1001
1400  CALL SUB3(POL    ,I,K)
      GO TO 1001
1500  K=I
      GO TO 1001
1001  CONTINUE
      GO TO 1100
```

Figure 6 (Cont'd)

```
C          ENTRY TO DERIVATIVE ROUTINE.
C          (ENTRANCE) AND (TERMINUS) MUST BE SET TO CORRECT VALUE PRIOR TO ENTRY.

    1 NDX=0
      REWIND 15

    2 IF (NDX .GT. 0) 500, 3
    3 NP=ENTRANCE
    4 NDX = NDX + 1
C          INITIALIZE COUNTERS
      ND=0 $ NE=0 $ NL=0 $ NS=0 $ NV=0 $ NX=1 $
      NP = NP + 1
    5 NV = NV + 1
      VARS(NV) = POL(NP)
      NP = NP + 1
      IF (POL(NP) .EQ. 2H.. ) 6, 5
    6 DO 619 NIV=NX,NV
      V = VARS(NIV)
    7 NO = 0
    8 NP = NP + 1
      IF (POL(NP) .EQ. 1H ) 8,9
    9 DO 10 N=1,6
      IF(POL(NP) . EQ. OP(N))12,10
   10 CONTINUE
      DO 11 N=1,2
      IF (POL(NP) .EQ. DUALOP(N))13,11
   11 CONTINUE
      NO = NO + 1
      NS = NS + 1
      SHU(NS) = POL(NP)
      GO TO 8
   12 IF (NO - 1) 300,200,100
   13 GO TO (2, 600) N
```

Figure 6 (Cont'd)

```
C 100     A B *
  100 NL=NL+1
      SING(NL)=NO-2
  101 RHO = SHU(NS)
      NS=NS-1
      LAM = SHU(NS)
      NS=NS-1
      IF(LAM .EQ. V) 102, 106
  102 LAMP = ONE
      IF(RHO . EQ. V ) 103, 105
  103 RHOP = ONE
      LR = 3
      GO TO 109
  105 RHOP = ZERO
      LR = 1
      GO TO 109
  106 LAMP = ZERO
      IF (RHO . EQ. V) 107,108
  108 RHOP = ZERO
      LR = 4
      ND=ND+1 $ D(ND)=ZERO
      BEGIND(NL)=ND
      GO TO 190
  107 RHOP = ONE
      LR = 2

  109 ND=ND+1
      BEGIND(NL)=ND
      GO TO (110,120,130,140,150,160) N


  110 GO TO (111,111,113) LR
  111 D(ND)=ONE
      GO TO 190
  113 D(ND)=TWO
      GO TO 190

  120 GO TO (111,122,123) LR
  122 D(ND)=ZERO
      ND=ND+1 $ D(ND)=ONE
      ND=ND+1 $ D(ND)=MINUS
      GO TO 190
```

Figure 6 (Cont'd)

```
  123 D(ND) = ZERO
      GO TO 190

  130 GO TO (131,132,133) LR
  131 D(ND)=RHO
      GO TO 190
  132 D(ND)=LAM
      GO TO 190
  133 D(ND)=TWO
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=MUL
      GO TO 190

  140 GO TO (141,142,123) LR
  141 D(ND)=ONE
      ND=ND+1 $ D(ND)=RHO
      ND=ND+1 $ D(ND)=DIV
      GO TO 190
  142 D(ND)=ZERO
      ND=ND+1 $ D(ND)=LAM
      ND=ND+1 $ D(ND)=MINUS
      ND=ND+1 $ D(ND)=RHO
      ND=ND+1 $ D(ND)=DIV
      ND=ND+1 $ D(ND)=RHO
      ND=ND+1 $ D(ND)=DIV
      GO TO 190

  150 GO TO (151,152,153) LR
C 151              DX(V..   V C **) =  C V C 1 - **   *
  151 D(ND)=RHO
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=RHO
      ND=ND+1 $ D(ND)=FIXONE
      ND=ND+1 $ D(ND)=MINUS
      ND=ND+1 $ D(ND)=POWER
      ND=ND+1 $ D(ND)=MUL
      GO TO 190
  152 D(ND)=LAM
C 152           DX(V..   C  V  ** )  =  C  V  **  LOGF  C  .  *
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=POWER
```

Figure 6 (Cont'd)

```
      ND=ND+1 $ D(ND)=4HLOGF
      ND=ND+1 $ D(ND)=LAM
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=MUL
      GO TO 190
C 153             DX(V..  V V **) = V V ** 1 LOGF V . + *
  153 D(ND)=V
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=POWER
      ND=ND+1 $ D(ND)=ONE
      ND=ND+1 $ D(ND)=4HLOGF
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=PLUS
      ND=ND+1 $ D(ND)=MUL
      GO TO 190

  160 DO 168 NN=1,6
      IF(LAM .EQ. FUNCTION (NN)) 169,168
  168 CONTINUE
      GO TO 705

  169 GO TO (161,162,163,164,165,166) NN

C 161             DX(V..  LOGF V .) =  1 V /
  161 D(ND)=ONE
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=DIV
      GO TO 190
C 162             DX(V..  EXPF V .) =  EXPF V .
  162 D(ND)=4HEXPF
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=FUNC
      GO TO 190
C 163             DX(V..  SINF V .) =  COSF V .
  163 D(ND)=4HCOSF
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=FUNC
      GO TO 190
C 164             DX(V..  COSF V .) =  0 SIN V . -
  164 D(ND)=ZERO
      ND=ND+1 $ D(ND)=4HSINF
```

Figure 6 (Cont'd)

```
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=MINUS
      GO TO 190
C 165           DX(V..  TANF V .) =  SECF V . SECF V . *
  165 D(ND)=4HSECF
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=4HSECF
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=MUL
      GO TO 190
C 166           DX(V.. SQRTF V .) =  .5 SQRTF V . /
  166 D(ND)=2H.5
      ND=ND+1 $ D(ND)=5HSQRTF
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=DIV

  190 ENDD(NL)=ND

C         FUNCTION STRING
  191 NE=NE+1 $ E(NE)=LAM
      BEGINE(NL)=NE
      NE=NE+1 $ E(NE)=RHO
      NE=NE+1 $ E(NE)=OP(N)

      ENDE(NL)=NE
      GO TO 7
```

Figure 6 (Cont'd)

```
C 200      LAM  B  *
  200 RHO=SHU(NS)
      NS=NS-1
      IF(RHO .EQ. V) 201,204
  201 RHOP=ONE
      IF(D(ND) .EQ. ZERO) 203,202
  202 LR=3
      GO TO 209
  203 LR=2
      GO TO 209
  204 IF(D(ND) .EQ.ZERO) 291,205
  205 LR=1
  209 GO TO (210,220,230,240,250,707)N
  210 GO TO (290,212,213)LR
  212 D(ND)=ONE
      GO TO 290
  213 ND=ND+1 $ D(ND)=ONE
      ND=ND+1 $ D(ND)=OP(N)
      GO TO 290

  220 GO TO (290,222,113)LR
  222 ND=ND+1 $ D(ND)=ONE
      ND=ND+1 $ D(ND)=MINUS
      GO TO 290
  230 GO TO (231,232,233)LR
  231 ND=ND+1 $ D(ND)=RHO
      ND=ND+1 $ D(ND)=MUL
      GO TO 290
  232 ND=ND-1
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 2321 J=K,L
      ND=ND+1
 2321 D(ND)=E(J)
      GO TO 290
  233 ND=ND+1 $ D(ND)=RHO
      ND=ND+1 $ D(ND)=MUL
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 234 J=K,L
      ND=ND+1
  234 D(ND)=E(J)
      ND=ND+1 $ D(ND)=PLUS
      GO TO 290
```

Figure 6 (Cont'd)

```
  240 GO TO (241,242,243)LR
  241 ND=ND+1 $ D(ND)=RHO
      ND=ND+1 $ D(ND)=DIV
      GO TO 290
  242 K=BEGINE(NL) $ L=ENDE(NL)
      DO 2421 J=K,L
      ND=ND+1
 2421 D(ND)=E(J)
      ND=ND+1 $ D(ND)=MINUS
      ND=ND+1 $ D(ND)=RHO
      ND=ND+1 $ D(ND)=DIV
      ND=ND+1 $ D(ND)=RHO
      ND=ND+1 $ D(ND)=DIV
      GO TO 290
  243 K=BEGINE(NL) $ L=ENDE(NL)
      DO 244 J=K,L
      ND=ND+1
  244 D(ND)=E(J)
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=DIV
      ND=ND+1 $ D(ND)=MINUS
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=DIV
      GO TO 290

  250 GO TO (251,252,253)LR
C 251             DX(V..  LAM C **) =  LAMP C * LAM  C 1 - **   *
  251 ND=ND+1 $ D(ND)=RHO
      ND=ND+1 $ D(ND)=MUL
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 2511 J=K,L
      ND=ND+1
 2511 D(ND)=E(J)
      ND=ND+1 $ D(ND)=RHO
      ND=ND+1 $ D(ND)=FIXONE
      ND=ND+1 $ D(ND)=MINUS
      ND=ND+1 $ D(ND)=POWER
      ND=ND+1 $ D(ND)=MUL
      GO TO 290
C 252             DX(V..  LAM C **) =  LAM V ** LOGF LAM . *
  252 ND=ND-1
```

Figure 6 (Cont'd)

```
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 2521 J=K,L
      ND=ND+1
 2521 D(ND)=E(J)
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=POWER
      ND=ND+1 $ D(ND)=4HLOGF
      DO 2522 J=K,L
      ND=ND+1
 2522 D(ND)=E(J)
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=MUL
      GO TO 290
C 253            DX(V..  LAM V **) =  LAMP V * LAM / LOGF LAM . + LAM V **    *
  253 ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=MUL
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 2531 J=K,L
      ND=ND+1
 2531 D(ND)=E(J)
      ND=ND+1 $ D(ND)=DIV
      ND=ND+1 $ D(ND)=4HLOGF
      DO 2532 J=K,L
      ND=ND+1
 2532 D(ND)=E(J)
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=PLUS
      DO 2533 J=K,L
      ND=ND+1
 2533 D(ND)=E(J)
      ND=ND+1 $ D(ND)=DIV
      ND=ND+1 $ D(ND)=POWER
      ND=ND+1 $ D(ND)=MUL

  290 ENDD(NL)=ND
C           FUNCTION STRING
  291 NE=NE+1 $ E(NE)=RHO
      NE=NE+1 $ E(NE)=OP(N)
      ENDE(NL)=NE
      GO TO 7
```

Figure 6 (Cont'd)

```
300 IF (SING(NL) .EQ. 0) 400, 301

C 301     A  RHO  *
  301 SING(NL)=SING(NL)-1
  302 LAM=SHU(NS)
      NS=NS-1
      IF(LAM .EQ. V) 303,306
  303 LAMP=ONE
      IF(D(ND) .EQ. ZERO) 305,304
  304 LR=3
      GO TO 309
  305 LR=1
      GO TO 309
  306 LAMP=ZERO
      IF(D(ND) .EQ. ZERO) 391,307
  307 LR=2

  309 GO TO (310,320,330,340,350,360) N

  310 GO TO (311,390,313) LR
  311 D(ND)=ONE
      GO TO 390
  313 D(ND)=ONE
      ND=ND+1 $ D(ND)=OP(N)
      GO TO 390

  320 GO TO (321,322,313)LR
  321 ND=ND+1 $ D(ND)=ONE
      ND=ND+1 $ D(ND)=MINUS
      GO TO 390
  322 K=ENDD(NL) $ L=K+1
      LENGTH = ENDD(NL) - BEGIND(NL) + 1
      DO 3221 J=1,LENGTH
      D(L)=E(K)
      K=K-1
 3221 L=L-1
      D(L)=ZERO
      ND=ND+2 $ D(ND)=MINUS
      GO TO 390

  330 GO TO (331,332,333)LR
  331 ND=ND-1
```

Figure 6 (Cont'd)

```
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 3311 J=K,L
      ND=ND+1
 3311 D(ND)=E(J)
      GO TO 390
  332 ND=ND+1 $ D(ND)=LAM
      ND=ND+1 $ D(ND)=MUL
      GO TO 390
  333 ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=MUL
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 334 J=K,L
      ND=ND+1
  334 D(ND)=E(J)
      ND=ND+1 $ D(ND)=PLUS
      GO TO 390

  340 GO TO (341,342,343)LR
  341 D(ND)=ONE
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 3411 J=K,L
      ND=ND+1
 3411 D(ND)=E(J)
      ND=ND+1 $ D(ND)=DIV
      GO TO 390
  342 ND=ND+1 $ D(ND)=ZERO
      ND=ND+1 $ D(ND)=LAM
      ND=ND+1 $ D(ND)=MINUS
      ND=ND+1 $ D(ND)=MUL
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 3422 JJ=1,2
      DO 3421 J=K,L
      ND=ND+1
 3421 D(ND)=E(J)
      ND=ND+1
 3422 D(ND)=DIV
      GO TO 390

  343 ND=ND+1 $ D(ND)=ZERO
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=MINUS
      ND=ND+1 $ D(ND)=MUL
```

Figure 6 (Cont'd)

```
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 344 J=K,L
      ND=ND+1
  344 D(ND)=E(J)
      ND=ND+1 $ D(ND)=PLUS
      DO 346 JJ=1,2
      DO 345 J=K,L
      ND=ND+1
  345 D(ND)=E(J)
      ND=ND+1
  346 D(ND)=DIV
      GO TO 390

  350 GO TO (351,352,353) LR
C 351             DX(V..  C RHO **) =  RHO V RHO 1 - ** *
  351 ND=ND-1
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 3511 J=K,L
      ND=ND+1
 3511 D(ND)=E(J)
      ND=ND+1 $ D(ND)=V
      DO 3512 J=K,L
      ND=ND+1
 3512 D(ND)=E(J)
      ND=ND+1 $ D(ND)=FIXONE
      ND=ND+1 $ D(ND)=MINUS
      ND=ND+1 $ D(ND)=POWER
      ND=ND+1 $ D(ND)=MUL
      GO TO 390
C 352             DX(V..  C RHO **) =  RHOP LOGF C . * C RHO **   *
  352 ND=ND+1 $ D(ND)=4HLOGF
      ND=ND+1 $ D(ND)=LAM
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=MUL
      ND=ND+1 $ D(ND)=LAM
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 3521 J=K,L
      ND=ND+1
 3521 D(ND)=E(J)
      ND=ND+1 $ D(ND)=POWER
      ND=ND+1 $ D(ND)=MUL
      GO TO 390
```

Figure 6 (Cont'd)

```
C 353              DX(V..  V RHO **) =  RHOP LOGF V . * RHO V / + V  RHO **   *
  353 ND=ND+1 $ D(ND)=4HLOGF
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=MUL
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 3531 J=K,L
      ND=ND+1
 3531 D(ND)=E(J)
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=DIV
      ND=ND+1 $ D(ND)=PLUS
      ND=ND+1 $ D(ND)=V
      DO 3532 J=K,L
      ND=ND+1
 3532 D(ND)=E(J)
      ND=ND+1 $ D(ND)=POWER
      ND=ND+1 $ D(ND)=MUL
      GO TO 390

  360 DO 368 NN=1,6
      IF (LAM .EQ. FUNCTION(NN)) 369,368
  368 CONTINUE
      GO TO 705
  369 GO TO (361,362,363,364,365,366) NN

C 361             DX(V..   LOGF RHO .) =  RHOP  RHO  /
  361 K=BEGINE(NL) $ L=ENDE(NL)
      DO 3611 J=K,L
      ND=ND+1
 3611 D(ND)=E(J)
      ND=ND+1 $ D(ND)=DIV
      GO TO 390
C 362             DX(V..   EXPF RHO .) =  RHOP  EXPF  RHO  .  *
  362 ND=ND+1 $ D(ND)=4HEXPF
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 3621 J=K,L
      ND=ND+1
 3621 D(ND)=E(J)
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=MUL
      GO TO 390
```

Figure 6 (Cont'd)

```
C 363               DX(V..   SINF RHO .) =  RHOP  COSF  RHO  .  *
  363 ND=ND+1 $ D(ND)=4HCOSF
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 3631 J=K,L
      ND=ND+1
 3631 D(ND)=E(J)
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=MUL
      GO TO 390
C 364               DX(V..   COSF RHO .) =  RHOP  0 SINF RHO . - *
  364 ND=ND+1 $ D(ND)=ZERO
      ND=ND+1 $ D(ND)=4HSINF
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 3641 J=K,L
      ND=ND+1
 3641 D(ND)=E(J)
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=MINUS
      ND=ND+1 $ D(ND)=MUL
      GO TO 390
C 365               DX(V..   TANF RHO .) =  RHOP  SECF  RHO  .  *  SECF  RHO  .  *
  365 DO 3652 JJ=1,2
      ND=ND+1 $ D(ND)=4HSECF
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 3651 J=K,L
      ND=ND+1
 3651 D(ND)=E(J)
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1
 3652 D(ND)=MUL
      GO TO 390
C 366               DX(V..  SQRTF RHO .) =  RHOP  .5  SQRTF V . / *
  366 ND=ND+1 $ D(ND)=2H.5
      ND=ND+1 $ D(ND)=5HSQRTF
      ND=ND+1 $ D(ND)=V
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=DIV
      ND=ND+1 $ D(ND)=MUL

  390 ENDD(NL)=ND

C          FUNCTION STRING    A  RHO  *
```

Figure 6 (Cont'd)

```
C 391     RELOCATE RHO UP ONE POSITION, AND THEN INSERT LAM.
      K=ENDE(NL)
  391 K=ENDE(NL) $ L=K+1
      LENGTH=ENDE(NL)-BEGINE(NL)+1
      DO 392 J=1,LENGTH
      E(L)=E(K)
      K=K-1
  392 L=L-1
      E(L)=LAM
C         PLACE CURRENT OPERATOR ON TOP.
      NE=NE+2 $ E(NE)=OP(N)

      ENDE(NL)=NE
      GO TO 7
```

Figure 6 (Cont'd)

```
C 400      LAM  RHO  *
  400 NL=NL-1
  401 IF(D(ENDD(NL)) .EQ. ZERO) 405,402
  402 IF(D(ENDD(NL+1)) .EQ. ZERO) 404,403
  403 LR=3
      GO TO 409
  404 LR=1
      GO TO 409
  405 IF(D(ENDD(NL+1)) .EQ. ZERO) 407,406
  407 ND=ND-1
      GO TO 490
  406 LR=2
  409 GO TO (410,420,430,440,450,707)N

C 410             DX(V..   LAM RHO -) = LAMP RHOP -
  410 GO TO (411,412,413) LR
  411 ND=ND-1
      GO TO 490
  412 D(ENDD(NL)) =1H
      GO TO 490
  413 ND=ND+1 $ D(ND)=OP(N)
      GO TO 490

  420 GO TO (411,413,413)LR

C 430             DX(V..   LAM RHO *) = LAMP RHO *  RHOP LAM *  +
  430 GO TO (431,432,433)LR
  431 ND=ND-1
      K=BEGINE(NL+1) $ L=ENDE(NL+1)
      DO 4311 J=K,L
      ND=ND+1
 4311 D(ND)=E(J)
      ND=ND+1 $ D(ND)=MUL
      GO TO 490
  432 D(ENDD(NL))=1H
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 4321 J=K,L
      ND=ND+1
 4321 D(ND)=E(J)
      ND=ND+1 $ D(ND)=MUL
      GO TO 490
C 433      RELOCATE RHOP UP SUFFICIENT POSITIONS TO PERMIT INSERTAION OF (RHO *).
```

Figure 6 (Cont'd)

```
C            K = OLD POSITION OF EACH ELEMENT OF RHOP.
C            L = NEW POSITION OF EACH ELEMENT OF RHOP.
  433 INSERT=ENDE(NL+1)-BEGINE(NL+1)+2
      K=ENDD(NL+1) $ L=K+INSERT
      ND=L
      LENGTH=ENDD(NL+1)-ENDD(NL)
      DO 4331 J=1,LENGTH
      D(L)=D(K)
      K=K-1
 4331 L=L-1
C          INSERT (RHO *) BETWEEN LAMP + RHOP.
      M=ENDD(NL)
      K=BEGINE(NL+1) $ L=ENDE(NL+1)
      DO 4332 J=K,L
      M=M+1
 4332 D(M)=E(J)
      M=M+1 $ D(M)=MUL
C          PLACE (LAM * +) ON TOP
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 4333 J=K,L
      ND=ND+1
 4333 D(ND)=E(J)
      ND=ND+1 $ D(ND)=MUL
      ND=ND+1 $ D(ND)=PLUS
      GO TO 490

  440 GO TO (441,442,443)LR
  441 ND=ND-1
      K=BEGINE(NL+1) $ L=ENDE(NL+1)
      DO 4411 J=K,L
      ND=ND+1
 4411 D(ND)=E(J)
      ND=ND+1 $ D(ND)=DIV
      GO TO 490
  442 K=BEGINE(NL) $ L=ENDE(NL)
      DO 4421 J=K,L
      ND=ND+1
 4421 D(ND)=E(J)
      ND=ND+1 $ D(ND)=MUL
      ND=ND+1 $ D(ND)=MINUS
      K=BEGINE(NL+1) $ L=ENDE(NL+1)
      DO 4423 JJ=1,2
```

Figure 6 (Cont'd)

```
      DO 4422 J=K,L
      ND=ND+1
 4422 D(ND)=E(J)
      ND=ND+1
 4423 D(ND)=DIV
C 443           DX(V..   LAM RHO /) = LAMP LAM RHO / RHOP * - RHO /
C         RELOCATE RHOP UP SUFFICIENT POSITIONS TO PERMIT INSERTION OF LAM RHO 1
  443 INSERT=ENDE(NL+1)-BEGINE(NL)+2
      K=ENDD(NL+1) $ L=K+INSERT
      ND=L
      LENGTH=ENDD(NL+1)-ENDD(NL)
      DO 4431 J=1,LENGTH
      D(L)=D(K)
      K=K-1
 4431 L=L-1
      K=BEGINE(NL) $ L=ENDE(NL+1)
      M=ENDD(NL)
      DO 4432 J=K,L
      M=M+1
 4432 D(M)=E(J)
C         THEN PLACE (* - RHO /) ON TOP.
      ND=ND+1 $ D(ND)=MUL
      ND=ND+1 $ D(ND)=MINUS
      K=BEGINE(NL+1)
      DO 4433 J=K,L
      ND=ND+1
 4433 D(ND)=E(J)
      ND=ND+1 $ D(ND)=DIV
      GO TO 490

  450 GO TO (451,452,453)LR
C 451     DX(V.. LAM RHO **) = LAMP RHO * LAM RHO 1 - ** *
C         RHO = 0. BACK-UP ONE SPACE
  451 ND=ND-1
      K=BEGINE(NL+1) $ L=ENDE(NL+1)
      DO 4511 J=K,L
      ND=ND+1
 4511 D(ND)=E(J)
      ND=ND+1 $ D(ND)=MUL
      K=BEGINE(NL)
      DO 4512 J=K,L
      ND=ND+1
```

Figure 6 (Cont'd)

```
 4512 D(ND)=E(J)
      ND=ND+1 $ D(ND)=FIXONE
      ND=ND+1 $ D(ND)=MINUS
      ND=ND+1 $ D(ND)=POWER
      ND=ND+1 $ D(ND)=MUL
      GO TO 490
C 452     DX(V.. LAM RHO **) = RHOP LAM RHO ** * LOGF LAM . *
  452 K=BEGINE(NL) $ L=ENDE(NL+1)
      DO 4521 J=K,L
      ND=ND+1
 4521 D(ND)=E(J)
      ND=ND+1 $ D(ND)=POWER
      ND=ND+1 $ D(ND)=MUL
      ND=ND+1 $ D(ND)=4HLOGF
      L=ENDE(NL)
      DO 4522 J=K,L
      ND=ND+1
 4522 D(ND)=E(J)
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=MUL
      GO TO 490

C 453     DX(V.. LAM RHO **) = LAMP RHO * LAM / RHOP LOGF LAM . * + LAM RHO ** *
C         RELOCATE RHOP UP SUFFICIENT POSITIONS TO INSERT (RHO * LAM /).
  453 INSERT=ENDE(NL+1)-BEGINE(NL)+3
      K=ENDD(NL+1) $ L=K+INSERT
      ND=L
      LENGTH=ENDD(NL+1)-ENDD(NL)
      DO 4531 J=1,LENGTH
      D(L)=D(K)
      K=K-1
 4531 L=L-1
C         INSERT (RHO * LAM /) BETWEEN (LAMP) AND (RHOP).
      K=BEGINE(NL+1) $ L=ENDE(NL+1)
      M=ENDD(NL)
      DO 4532 J=K,L
      M=M+1
 4532 D(M)=E(J)
      M=M+1 $ D(M)=MUL
      K=BEGINE(NL) $ L=ENDE(NL)
      DO 4533 J=K,L
      M=M+1
```

Figure 6 (Cont'd)

```
 4533 D(M)=E(J)
      ND=ND+1 $ D(ND)=DIV
C         THEN PLACE (LOGF LAM . * + LAM RHO **  *) ON TOP.
      ND=ND+1 $ D(ND)=4HLOGF
      DO 4534 J=K,L
      ND=ND+1
 4534 D(ND)=E(J)
      ND=ND+1 $ D(ND)=FUNC
      ND=ND+1 $ D(ND)=MUL
      ND=ND+1 $ D(ND)=PLUS
      L=ENDE(NL+1)
      DO 4535 J=K,L
      ND=ND+1
 4535 D(ND)=E(J)
      ND=ND+1 $ D(ND)=POWER
      ND=ND+1 $ D(ND)=MUL

  490 ENDD(NL)=ND

C         FUNCTION STRING
      NE=NE+1 $ E(NE)=OP(N)
      ENDE(NL)=NE
      GO TO 7
```

Figure 6 (Cont'd)

```
  500 WRITE TAPE 15,  ND,NE,NL,NP,NS,NV,NIV,D,E,V,VARS,
     1                BEGIND,BEGINE,ENDD,ENDE, ENTRANCE,SHU,SING
      ENTRANCE = NP
      GO TO 4
C 600      POLISH(NP) IS THE DUAL-OPERATOR SUFFIX (,,)
  600 IF(NS .EQ. 1) 601, 605
  601 IF(ND .EQ. 0) 602, 703
  602 ND=1 $ NE=1
      E(1) = SHU(1)
      IF(SHU(1) .EQ. V) 603, 604
  603 D(1) = ONE $ GO TO 605
  604 D(1) = ZERO
  605 PRINT 606,(E(J), J=1,NE)
  606 FORMAT(//26H           EXPRESSION.      , 15A6, (//26X,15A6))
      PRINT 607, V, (D(J), J=1,ND)
  607 FORMAT(//17H            DE / D, A1, 8X,   15A6, (//26X,15A6))
      EXIT = NP
C         RELOCATE THE REMAINDER OF THE POLISH STRING TO THE RIGHT
C         OR LEFT SUFFICIENT POSITIONS TO PERMIT INSERTION OF THE
C         DERIVATIVE STRING BETWEEN THE (ENTRANCE) AND (EXIT) POSITIONS.
  610 NDIFF = ENTRANCE + ND - (EXIT - 1)
      LENGTH=TERMINUS-EXIT+1
      IF (NDIFF) 611, 615, 613
  611 K = EXIT
      L = EXIT + NDIFF
      DO 612 J=1,LENGTH
      POL(L) = POL(K)
      K=K+1
  612 L=L+1
      GO TO 615
  613 K = TERMINUS
      L = TERMINUS + NDIFF
      DO 614 J=1,LENGTH
      POL(L) = POL(K)
      K=K-1
  614 L=L-1
  615 TERMINUS = TERMINUS + NDIFF
      EXIT = EXIT + NDIFF
      NP=ENTRANCE
      DO 616 J=1,ND
      NP=NP+1
```

Figure 6 (Cont'd)

```
  616 POL(NP) = D(J)
      NP=ENTRANCE
      ND=0 $ NE=0 $ NL=0 $ NS=0 $
  619 CONTINUE
C 620     DIFFERENTIATION COMPLETED.  DELETE THE DUAL-OPERATOR DX FROM
C           THE POLISH STRING.
  620 POL(ENTRANCE) = 8H
      POL(EXIT)     = 8H
      NDX = NDX - 1
      IF (NDX .GT. 0) 621, 950

C 950     PROBLEM TERMINATES IF NDX .EQ. 0.

  621 BACKSPACE 15
  622 READ TAPE 15,  ND,NE,NL,NP,NS,NV,NIV,D,E,V,VARS,
     1               BEGIND,BEGINE,ENDD,ENDE,ENTRANCE,SHU,SING
      BACKSPACE 15
      NX=NIV
      GO TO 6

  703 PRINT 704
  704 FORMAT (/33H MACHINE ERROR. RESUBMIT PROGRAM.)
      GO TO 999
  705 PRINT 706
  706 FORMAT (/27H ERROR. UNDEFINED FUNCTION.)
      GO TO 999
  707 PRINT 708
  708 FORMAT (/42H ERROR. ARGUMENT NOT ASSIGNED TO FUNCTION.)
      GO TO 999

  950 CONTINUE

C         DIFFERENTIATION HAS BEEN COMPLETED.
C         REDUCE EVALUATION ROUTINE  POLISH STRING POSITION COUNTER.
C         RETURN TO  EVALUATION ROUTINE
      I=ENTRANCE-1
      GO TO 1001

1100  CALL SUB4(TEMP)

  999 CONTINUE
      END
```

Figure 6 (Cont'd)

```
      SUBROUTINE MICRO(POL    ,I)
C           THIS SUBROUTINE GENERATES THE MACROS FOR SYMBOLS IN THE
C            EQUATION
      TYPE INTEGER POL
      DIMENSION MI(10),POL    (1)
      MI(1)=3HINI
      MI(2)=1H1
      MI(3)=1H1
      MI(4)=3HLDA
      MI(5)=1H0
      MI(6)=POL    (I)
      MI(7)=3HSTA
      MI(8)=1H1
      MI(9)=4HTEMP
      MI(10)=0
      CALL MACRO(MI)
      CONTINUE
      END
```

Figure 6 (Cont'd)

```
      SUBROUTINE SUB1(POL    ,I)
C          THIS SUBROUTINE GENERATES THE MACROS FOR THE FOLLOWING
C           OPERATORS IN THE EQUATION,  + ,  - ,  * ,  /
      TYPE INTEGER POL
      DIMENSION MI(13),POL    (1)
      MI(1)=3HLDA
      MI(2)=1H1
      MI(3)=6HTEMP-1
      IF(POL    (I) .EQ. 1H+)5,1
1     IF(POL    (I) .EQ. 1H-)6,2
2     IF(POL    (I) .EQ. 1H*)7,3
3     IF(POL    (I) .EQ. 1H/)8,4
    4 PRINT 9
    9 FORMAT(18H YOU HAVE AN ERROR)
      GO TO 16
    5 MI(4)=3HFAD
      GO TO 10
    6 MI(4)=3HFSU
      GO TO 10
    7 MI(4)=3HFMU
      GO TO 10
    8 MI(4)=3HFDV
   10 MI(5)=1H1
      MI(6)=4HTEMP
      MI(7)=3HSTA
      MI(8)=1H1
      MI(9)=6HTEMP-1
      MI(10)=3HINI
      MI(11)=1H1
      MI(12)=2H-1
      MI(13)=0
      CALL MACRO(MI)
16    CONTINUE
      END
```

Figure 6 (Cont'd)

```
      SUBROUTINE SUB2(NUM)
C          THIS SUBROUTINE GENERATES MACROS FOR EXPONTIATION WHEN THERE
C         IS NO MATHAMATICAL PROCESS REQUIRED TO FIND THE POWER TO WHICH
C         THE SYMBOL IS BEING RAISED. IT IS FOR FIXED POINT ONLY
      DIMENSION MI(40)
      J=NUM
      J=J-1
      MI(1)=3HLDA
      MI(2)=1H1
      MI(3)=6HTEMP-1
      MI(4)=3HFMU
      MI(5)=1H1
      MI(6)=6HTEMP-1
      L=7
20    J=J-1
      IF(J)3,4,3
3     MI(L)=3HFMU
      MI(L+1)=1H1
      MI(L+2)=6HTEMP-1
      L=L+3
      GO TO 20
4     MI(L)=3HSTA
      MI(L+1)=1H1
      MI(L+2)=6HTEMP-1
      MI(L+3)=3HINI
      MI(L+4)=1H1
      MI(L+5)=2H-1
      MI(L+6)=0
      CALL MACRO(MI)
      CONTINUE
      END
```

Figure 6 (Cont'd)

```
      SUBROUTINE SUB3(POL    ,I,K)
C       THIS SUBROUTINE GENERATES MACROS FOR CALLING IN FUNCTION ROUTINES
C        WHENEVER THEY ARE ENCOUNTERED IN THE EQUATION
      TYPE INTEGER POL
      DIMENSION MI(10),POL    (1)
      MI(1)=3HLDA
      MI(2)=1H1
      MI(3)=4HTEMP
      MI(4)=3HRTJ
      MI(5)=1H0
      MI(6)=POL    (K)
      MI(7)=3HSTA
      MI(8)=1H1
      MI(9)=4HTEMP
      MI(10)=0
      CALL MACRO(MI)
      CONTINUE
      END
```

Figure 6 (Cont'd)

```
      SUBROUTINE SUB4(TEMP)
C         THIS SUBROUTINE GENERATES MACROS TO SET THE INDEX COUNTER
C          BACK TO ITS ORIGINAL COUNT PRIOR TO THE GENERATING OF ANY
C       MACROS
      DIMENSION MI(10)
      MI(1)=3HLDA
      MI(2)=1H1
      MI(3)=4HTEMP
      MI(4)=3HSTA
      MI(5)=1H0
      MI(6)=4HTEMP
      MI(7)=3HINI
      MI(8)=1H1
      MI(9)=2H-2
      MI(10)=0
      CALL MACRO(MI)
      CONTINUE
      END
```

Figure 6 (Cont'd)

```
      SUBROUTINE SUB5(I)
C         THIS SOUBROUTINE GENERATES MACROS FOR EXPONENTIATION WHEN
C        A MATHAMATICAL PROCESS MUST BE USED PRIOR TO RAISING A SYMBOL
C        TO A POWER
      DIMENSION MI(34)
      MI(1)=3HSIL   $   MI(2)=1H2   $   MI(3)=4HPART   $
      MI(4)=3HLDA   $   MI(5)=1H1   $   MI(6)=4HTEMP   $
      MI(7)=3HSTA   $   MI(8)=1H0    $   MI(9)=5HPARTI     $
      MI(10)=3HLIL   $   MI(11)=1H2   $   MI(12)=5HPARTI   $
      MI(13)=3HINI   $   MI(14)=1H2   $   MI(15)=2H-1   $
      MI(16)=3HLDA   $   MI(17)=1H1   $   MI(18)=6HTEMP-1   $
      MI(19)=3HFMU   $   MI(20)=1H1   $   MI(21)=6HTEMP-1   $
      MI(22)=3HIJP   $   MI(23)=1H2   $   MI(24)=MI(19)
      MI(25)=3HSTA   $   MI(26)=1H1   $   MI(27)=6HTEMP-1   $
      MI(28)=3HLIL   $   MI(29)=1H2   $   MI(30)=4HPART   $
      MI(31)=3HINI   $   MI(32)=1H1  $  MI(33)=2H-1  $   MI(34)=0  $
      CALL MACRO(MI)
      CONTINUE
      END
```

Figure 6 (Cont'd)

```
      SUBROUTINE MACRO(MI)
C           THIS SUBROUTINE PRINTS OUT THE MACROS WHICH WERE GENERATED
C           BY THE MACRO ROUTINES
      DIMENSION MI(1)
      I=1
    4 IF(MI(I))2,1,2
    2 PRINT 3,MI(I),MI(I+1),MI(I+2)
    3 FORMAT(3(A6))
      I=I+3
      GO TO 4
    1 RETURN
      END
         END
         FINIS
-EXECUTE.
```

Figure 6 (Cont'd)

```
C           TEST PROBLEMS
     1 Z=DX(X..X)
     2 Z=DX(X..X+C)
     3 Z=DX(X..X**X)
     4 Z=DX(X..X**(X+2 ))
     5 Z=DX(X..X**2 )
     6 Z=DX(X..X**(2 *A))
     7 Z=DX(X..X*Y)
     8 Z=DX(X..X/(2.*X))
     9 Z=DX(X..SINF(X))
    10 Z=DX(X..COSF(X))
    11 Z=DX(X..TANF(X))
    12 Z=DX(X..LOGF(X))
    13 Z=DX(X..EXPF(X))
    14 Z=DX(X..SQRTF(X))
    15 Z=DX(X..SINF(X+2.))
    16 Z=DX(X..SINF(X**2 +2.*X))
    17 Z=DX(X..COSF(X*2.))
    18 Z=DX(X..(X**2 )*(X**2 ))
    19 Z=DX(X..(X+X**2 )-(X+2.)**2 )
    20 Z=DX(X..(SINF(X**2 ))**2 )
    21 Z=DX(X..SINF(LOGF(X)))
    22 Z=DX(X..EXPF(X**2 +X))
    23 Z=DX(X..2.*X**2 +4*X)
    24 Y=DX(X..TANF(X+2.)+DX(X..SINF(X**2 )))
    25 Y=DX(X..TANF(X+2.))+DX(X..SINF(X**2 ))
    26 Z=DX(X,Y..X**(Y**2 )+X*Y)
    27 Z=DX(X..X*(SINF(Y*X**2 ))**2 )
    28 W=DX(X,Y,Z..X**4 *Y+2.*X**3 *Y**2 *Z+3.*X**2 *Y**3 *Z**2 -DX(X,Y..
      14.*X*Y**4 *Z**3 ))
    29 A = DX(X..X*2. + DX(X..X**2  + DX(X..X**3)))
   100  Z=DX(I..(G*SINF(I)**2 + H * COSF(I)**2  + F + ((F**2  +  2. * F  *
      1     (G * SINF(I)**2 + H * COSF(I)**2) + G**2 * SINF(I)**2  +
      2     H**2*COSF(I)**2) * (1.+ 2. * T *  (G * SINF(I)**2  +  H *
      3     COSF(I)**2) + T**2 * (G**2 * SINF(I)**2 + H**2 * COSF(I)**2)))
      4     **(1/2) + T * (F * (G * SINF(I)**2 + H * COSF(I)**2)  +
      5     G**2 * SINF(I)**2 + H**2 * SINF(I)**2)) / ((1. - F * T) *
      6     (G * SINF(I)**2 + H * COSF(I)**2 + F + ((F**2 + 2. * F  *
      7     (G * SINF(I)**2 + H * COSF(I)**2)  +  G**2  * SINF(I)**2  +
      8     H**2  *  COSF(I)**2)  **(1/2)  ))))
```

Figure 7

```
FORTRAN.        1 Z=DX(X..X)
                  $

POLISH.           Z      DX     X      ..     X      ,,     =

EXPRESSTON.       X

 DE / DX          1.

FORTRAN.        2 Z=DX(X..X+C)
                  $

POLISH.           Z      DX     X      ..     X      C      +      ,,     =

EXPRESSION.       X      C      +

 DE / DX          1.

FORTRAN.        3 Z=DX(X..X**X)
                  $

POLISH.           Z      DX     X      ..     X      X      **     ,,     =

EXPRESSION.       X      X      **

 DE / DX          X      X      **     1.     LOGF   X      .      +      *
```

Figure 7 (Cont'd)

```
FORTRAN.        4 Z=DX(X..X**(X+2 ))
                  $

POLISH.           Z     DX    X     ..    X     X     2     +     **    ,,    =

EXPRESSION.       X     X     2     +     **

 DE / DX          1.    LOGF  X     .     *     X     2     +     X     /     +     X     X     2     +
                  **    *

FORTRAN.        5 Z=DX(X..X**2 )
                  $

POLISH.           Z     DX    X     ..    X     2     **    ,,    =

EXPRESSION.       X     2     **

 DE / DX          2     X     2     1     -     **    *

FORTRAN.        6 Z=DX(X..X**(2 *A))
                  $

POLISH.           Z     DX    X     ..    X     2     A     *     **    ,,    =

EXPRESSION.       X     2     A     *     **

 DE / DX          2     A     *     X     2.    A     *     1     -     **    *
```

Figure 7 (Cont'd)

```
FORTRAN.       16 Z=DX(X..SINF(X**2 +2.*X))
                  $

POLISH.           Z     DX    X     ..    SINF  X     2     **    2.    X     *     +     .     ,,    =

EXPRESSION.       SINF  X     2     **    2.    X     *     +     .

 DE / DX          2     X     2     1     -     **    *     2.    +     COSF  X     2     **    2.    X
                  *     +     .     *

FORTRAN.       17 Z=DX(X..COSF(X*2.))
                  $

POLISH.           Z     DX    X     ..    COSF  X     2.    *     .     ,,    =

EXPRESSION.       COSF  X     2.    *     .

 DE / DX          2.    0.    SINF  X     2.    *     .     -     *

FORTRAN.       18 Z=DX(X..(X**2 )*(X**2 ))
                  $

POLISH.           Z     DX    X     ..    X     2     **    X     2     **    *     ,,    =

EXPRESSION.       X     2     **    X     2     **    *

 DE / DX          2     X     2     1     -     **    *     X     2     **    *     2     X     2     1
                  -     **    *     X     2     **    *     +
```

Figure 7 (Cont'd)

```
FORTRAN.     24 Y=DX(X..TANF(X+2.)+DX(X..SINF(X**2 )))
                $

POLISH.         Y     DX    X     ..    TANF  X     2.    +     .     DX    X     ..    SINF  X     2
                **    .     ,,    +     ,,    =

EXPRESSION.     SINF  X     2     **    .

 DE / DX        2     X     2     1     -     **    *     COSF  X     2     **    .     *

EXPRESSION.     TANF  X     2.    +     .     2     X     2     1     -     **    *     COSF  X     2
                **    .     *     +

 DE / DX        1.    SECF  X     2.    +     .     *     SECF  X     2.    +     .     *     2     1
                -     X     2     1     -     1     -     **    *     2     *     COSF  X     2     *

                .     *     2     X     2     1     -     **    *     0.    SINF  X     2     **    .

                -     *     2     X     2     1     -     **    *     *     +     +

FORTRAN.     25 Y=DX(X..TANF(X+2.))+DX(X..SINF(X**2 ))
                $

POLISH.         Y     DX    X     ..    TANF  X     2.    +     .     ,,    DX    X     ..    SINF  X
                2     **    .     ,,    +     =

EXPRESSION.     TANF  X     2.    +     .

 DE / DX        1.    SECF  X     2.    +     .     *     SECF  X     2.    +     .     *

EXPRESSION.     SINF  X     2     **    .

 DE / DX        2     X     2     1     -     **    *     COSF  X     2     **    .     *
```

Figure 7 (Cont'd)